Instruction Set Architectures

Instruction Set Architectures

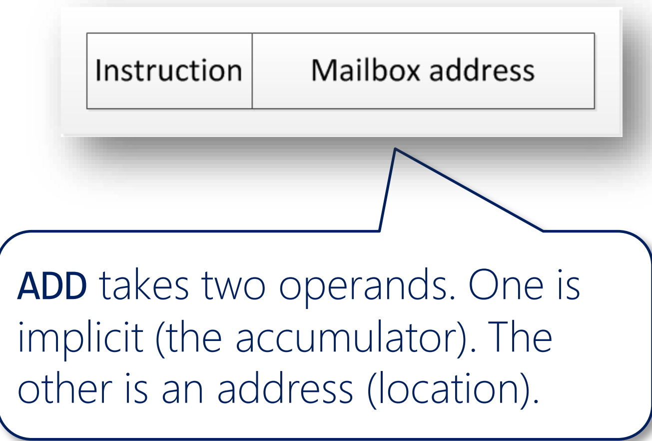
ISA determines instruction formats

- The LMC is a one-address architecture (an accumulator-based machine).

Instruction Set Architectures

ISA determines instruction formats

- The LMC is a one-address architecture (an accumulator-based machine).
  - e.g., the instruction ADD X



Instruction Set Architectures

ISA determines instruction formats

- There are other instruction set architectures, all based on the number of explicit operands.
  - 0-address (stack)
  - 1-address (accumulator)
  - 2-address
  - 3-address

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack

a

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack
b
a

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
→ PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack
a * b

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack
c
a * b

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
→ PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack
d
c
a * b

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
→ MUL	0
SUB	0
ADD	0
POP A	1

Stack
e
d
c
a * b

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
→ SUB	0
ADD	0
POP A	1

Stack
d * e
c
a * b

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
→ ADD	0
POP A	1

Stack
c – (d * e)
a * b

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack
a * b + (c – (d * e))

Instruction Set Architectures

0-Address Machines

- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e

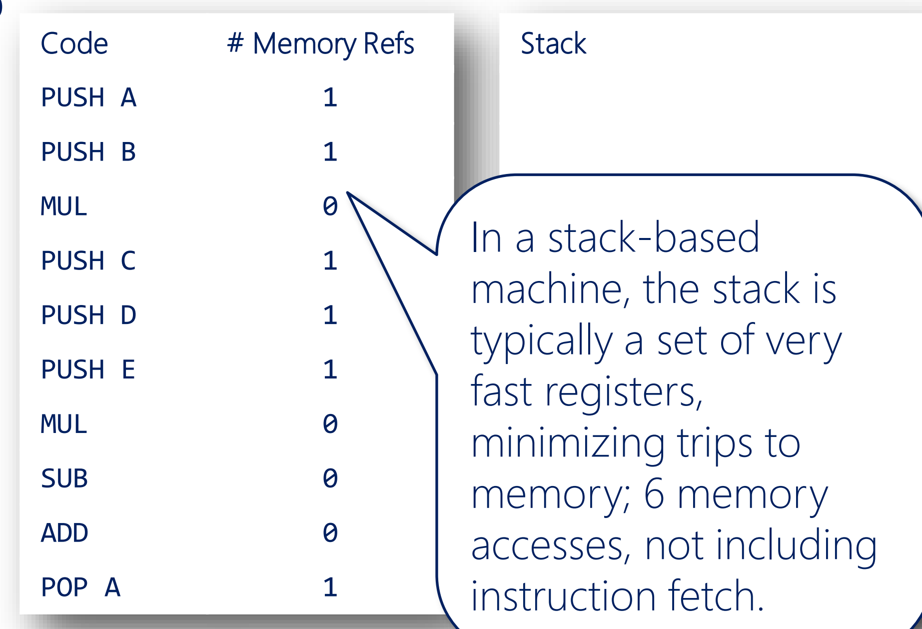
Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

Stack

Instruction Set Architectures

0-Address Machines

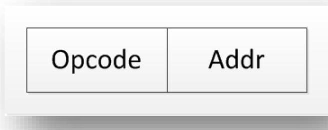
- All operands for binary operations are implicit on the stack. Only push/pop reference memory.
  - e.g., calculating
    a = a * b + c – d * e



Instruction Set Architectures

1-Address Machines

- Accumulator is a source and destination. Second source is explicit.



Instruction Set Architectures

1-Address Machines

- Accumulator is a source and destination. Second source is explicit.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
LOAD A	1
MUL B	1
ADD C	1
STORE T1	1
LOAD D	1
MUL E	1
STORE T2	1
LOAD T1	1
SUB T2	1
STORE A	1

Instruction Set Architectures

1-Address Machines

- Accumulator is a source and destination. Second source is explicit.
  - e.g., calculating
    a = a * b + c – d * e

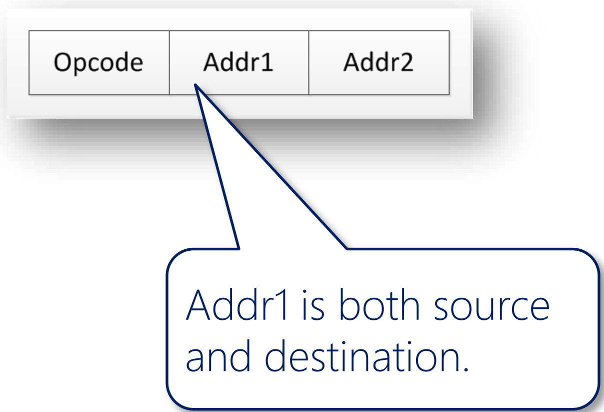
Code	# Memory Refs
LOAD A	1
MUL B	1
ADD C	1
STORE T1	1
LOAD D	1
MUL E	1
STORE T2	1
LOAD T1	1
SUB T2	1
STORE A	1

10 memory references, not including instruction fetch.

Instruction Set Architectures

2-Address Machines

– Two source addresses for operands.
  One source is also the destination.



Instruction Set Architectures

2-Address Machines

- Two source addresses for operands.
  One source is also the destination.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
MOVE T1, A	2
MUL T1, B	3
ADD T1, C	3
MOVE T2, D	2
MUL T2, E	3
SUB T1, T2	3
MOVE A, T1	2

Instruction Set Architectures

2-Address Machines

- Two source addresses for operands.
  One source is also the destination.
  - e.g., calculating
    a = a * b + c – d * e

Code	# Memory Refs
`MOVE T1, A`	2
`MUL T1, B`	3
`ADD T1, C`	3
`MOVE T2, D`	2
`MUL T2, E`	3
`SUB T1, T2`	3
`MOVE A, T1`	2

Using memory-to-memory operations, 18 memory accesses (not including instruction fetch). What if T1 and T2 were registers?

Instruction Set Architectures

2-Address Machines

- Two source addresses for operands. One source is also the destination.
  - e.g., calculating a = a * b + c – d * e

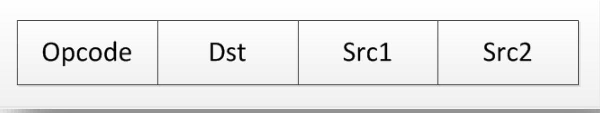
Code	# Memory Refs
MOVE R1, A	1
MUL R1, B	1
ADD R1, C	1
MOVE R2, D	1
MUL R2, E	1
SUB R1, T2	0
MOVE A, R1	1

Cuts memory references down to 6. This is called a 1½ address machine with a load/store architecture.

Instruction Set Architectures

3-Address Machines

- One destination operand,
  two source operands,
  all explicit



3-Address Machines

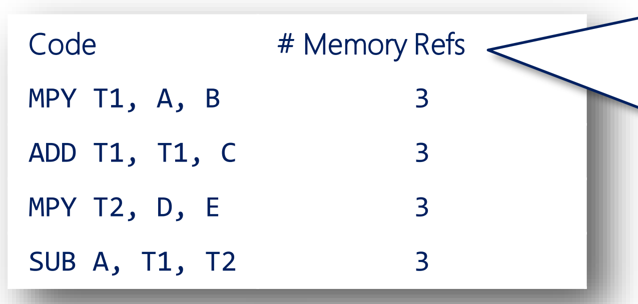
- One destination operand, two source operands, all explicit
  - e.g., calculating a = a * b + c – d * e

Code	# Memory Refs
MPY T1, A, B	3
ADD T1, T1, C	3
MPY T2, D, E	3
SUB A, T1, T2	3

Instruction Set Architectures

3-Address Machines

- One destination operand, two source operands, all explicit
  - e.g., calculating $a = a * b + c - d * e$



Code	# Memory Refs
`MPY T1, A, B`	3
`ADD T1, T1, C`	3
`MPY T2, D, E`	3
`SUB A, T1, T2`	3

12 memory accesses, not including instruction fetch. What if T1, T2 were registers?

Instruction Set Architectures

3-Address Machines

- One destination operand, two source operands, all explicit
  - e.g., calculating a = a * b + c – d * e

Code	# Memory Refs
`MPY R1, A, B`	2
`ADD R1, R1, C`	1
`MPY R2, D, E`	2
`SUB A, R1, R2`	1

6 memory accesses; general purpose registers make a substantial difference.

Instruction Set Architectures

Comparison

- Assume 8 registers (3 bits),
  32 op-codes (5 bits),
  15-bit addresses,
  16-bit integers.
- Which ISA accesses memory the least?

Instruction Set Architectures

Comparison

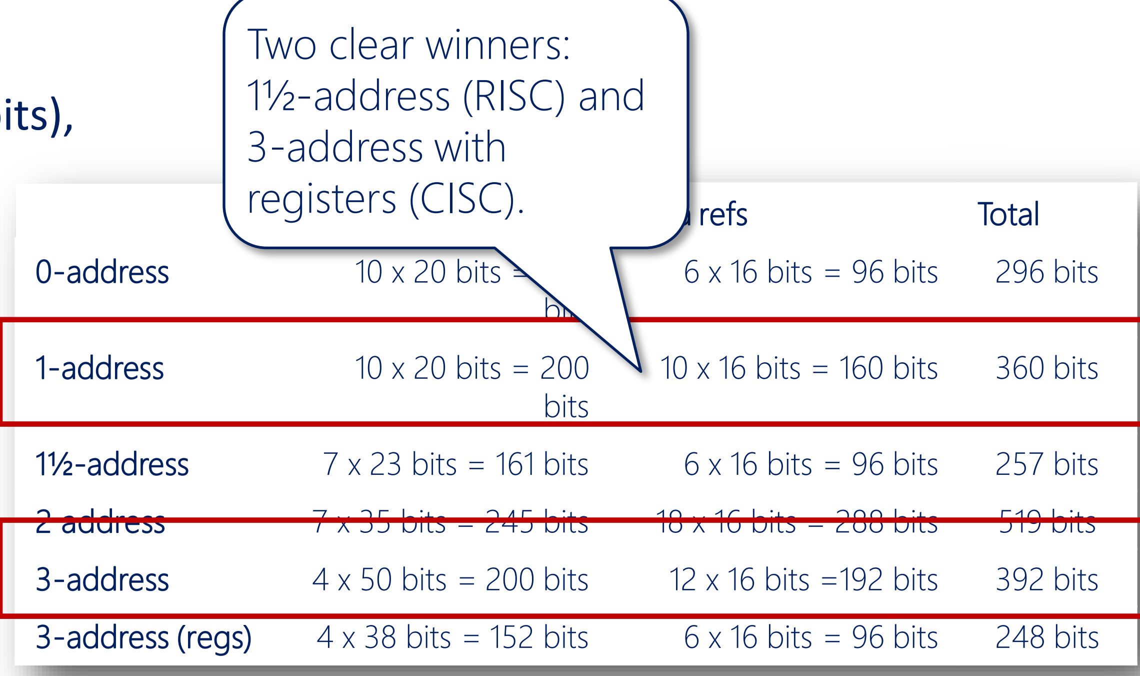
- Assume 8 registers (3 bits), 32 op-codes (5 bits), 15-bit addresses, 16-bit integers.
- Which ISA accesses memory the least?

	Instructions	Data refs	Total
0-address	10 x 20 bits = 200 bits	6 x 16 bits = 96 bits	296 bits
1-address	10 x 20 bits = 200 bits	10 x 16 bits = 160 bits	360 bits
1½-address	7 x 23 bits = 161 bits	6 x 16 bits = 96 bits	257 bits
2 address	7 x 35 bits = 245 bits	18 x 16 bits = 288 bits	519 bits
3-address	4 x 50 bits = 200 bits	12 x 16 bits =192 bits	392 bits
3-address (regs)	4 x 38 bits = 152 bits	6 x 16 bits = 96 bits	248 bits

Instruction Set Architectures

Comparison

- Assume 8 registers (3 bits), 32 op-codes (5 bits), 15-bit addresses, 16-bit integers.
- Which ISA accesses memory the least?



		refs	Total
0-address	10 x 20 bits = [illegible]	6 x 16 bits = 96 bits	296 bits
1-address	10 x 20 bits = 200 bits	10 x 16 bits = 160 bits	360 bits
1½-address	7 x 23 bits = 161 bits	6 x 16 bits = 96 bits	257 bits
2-address	7 x 35 bits = 245 bits	18 x 16 bits = 288 bits	519 bits
3-address	4 x 50 bits = 200 bits	12 x 16 bits =192 bits	392 bits
3-address (regs)	4 x 38 bits = 152 bits	6 x 16 bits = 96 bits	248 bits

Instruction Set Architectures

Summary

- The instruction set architecture determines the format of instructions (and therefore the assembly language).
- Four basic types with variations:
  - 0-address (stack)
  - 1-address (accumulator)
  - 2-address (register variant is 1½-address)
  - 3-address (with register variant)
- ISA dramatically affects the number of times memory is accessed.

Assembly Language

Assembly Language

Generations of programming languages

- First generation: programmed directly in binary using wires or switches.

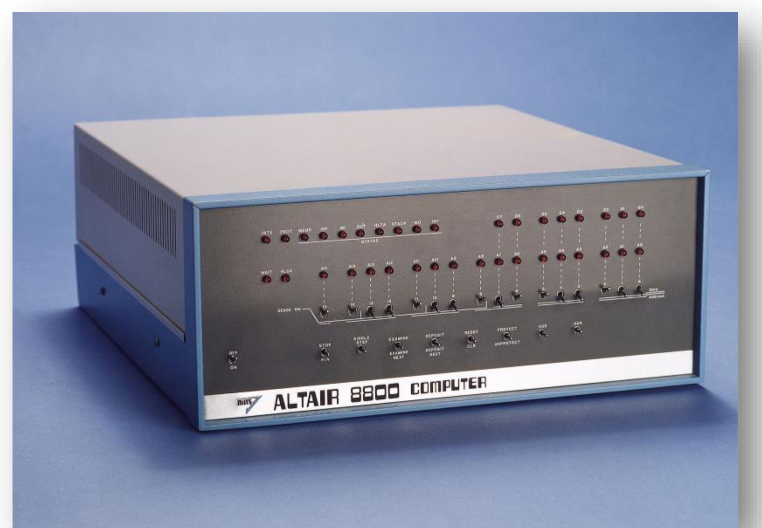


Image credit: http://professornerdster.com/wp-content/uploads/2012/04/altair_8800.jpeg

Assembly Language

Generations of programming languages

- Second generation: assembly language. Human readable, converted directly to machine code.

Image credit: http://www.coprolite.com/zhb/pooldog/grnscn.jpg

Assembly Language

Generations of programming languages

– Third generation: high-level languages, while loops, if-then-else, structured. Most programming today, including object-oriented.

```
 * Then all the LEDs are turned off and the cycle repea
 */
public class Sawtooth extends MIDlet {

    private ITriColorLED leds[] = EDemoBoard.getInstanc
    private ISwitch sw2 = EDemoBoard.getInstance().getS
    protected void startApp() throws MIDletStateChangeE
        System.out.println("Hello, sawtooth");
        while (sw2.isOpen()) {
            for (int i = 0; i < 8; i++) {
                int val = 32 * (i + 1) - 1;
                leds[i].setRGB(val, 0, val);
                leds[i].setOn();
                Utils.sleep(1000);
```

Image credit: http://www.sunspotworld.com/Tutorial/pngs/swemul06.png

Assembly Language

Generations of programming languages

– Fourth generation: 1990s natural languages, non-procedural, report generation. Use programs to generate other programs. Limited use today.

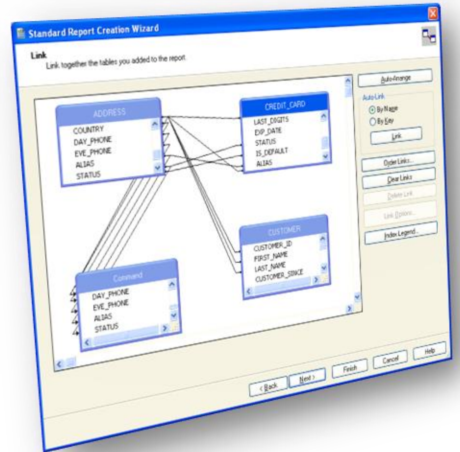


Image credit: http://docs.oracle.com/cd/E13167_01/aldsp/docs25/appdev/wwimages/CrystalReportsStep4.gif

Assembly Language

Generations of programming languages

- Key idea: Regardless of the language of writing, computers only process machine code.
- All non-machine code goes through a translation phase into machine code.
  - Code generators
  - Compilers
  - Assemblers

Assembly Language

Language translation process

- High level languages use comparison constructs, loops, variables, etc.
- Machine code is binary, directly executed by CPU.

How does this...

```
1  var i = 0;
2  var j = 1;
3  var k = 0;
4  while (k < 10) {
5    var fib = i + j;
6    i = j;
7    j = fib;
8    print(i);
9    k = k + 1
10 }
```

...become this?

```
7C40 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55
7C50 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E
7C60 26 66 68 00 00 00 00 66 FF 76 08 68 00
7C70 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B
7C80 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C
7C90 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61
7CA0 4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A
7CB0 EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C
7CC0 7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85
7CD0 D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00
7CE0 64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75
7CF0 FB 54 43 50 41 75 32 81 F9 02 01 72 2C
7D00 BB 00 00 66 68 00 02 00 00 66 68 08 00
7D10 53 66 53 66 55 66 68 00 00 00 00 66 68
7D20 00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA
7D30 00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03
7D40 32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB
7D50 0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02
7D60 02 C3 49 6E 76 61 6C 69 64 20 70 61 72
```

Assembly Language

Language translation process

- Convert high level language to if/goto.

```
1  var i = 0;
2  var j = 1;
3  var k = 0;
4  while (k < 10) {
5    var fib = i + j;
6    i = j;
7    j = fib;
8    print(i);
9    k = k + 1
10 }
```

Assembly Language

Language translation process

- Convert high level language to if/goto.

```
1  var i = 0;
2  var j = 1;
3  var k = 0;
4  while (k < 10) {
5    var fib = i + j;
6    i = j;
7    j = fib;
8    print(i);
9    k = k + 1
10 }
```

```
       i = 0
       j = 1
       k = 0
loop:  if (k – 10 == 0) goto done
       fib = i + j
       i = j
       j = fib
       print i
       k = k + 1
       goto loop
done:  halt
```

Use *labels* for branch targets.

Assembly Language

Language translation process

- Convert if/goto to assembly (LMC here).

```
loop: LDA  k    ; if (k - 10 == 0) goto done
      SUB  ten  ;
      BRZ  done ;
      LDA  i    ; fib = i + j
      ADD  j    ;
      STO  fib  ;
      LDA  j    ; i = j
      STO  i    ;
      LDA  fib  ; j = fib
      STO  j    ;
      LDA  i    ; print i
      OUT       ;
      LDA  k    ; k = k + 1
      ADD  one  ;
      STO  k    ;
      BR   loop ; goto loop
done: HLT       ; halt
```

Assembly Language

Language translation process

- Convert if/goto to assembly (LMC here).

```
loop: LDA  k    ; if (k - 10 == 0) goto done
      SUB  ten  ;
      BRZ  done ;
      LDA  i    ; fib = i + j
      ADD  j    ;
      STO  fib  ;
```

```
; data section
j:    DAT 0    ; i = 0
i:    DAT 1    ; j = 1
k:    DAT 0    ; k = 0
fib:  DAT 0    ;
ten:  DAT 10   ;
one:  DAT 1    ;
```

```
      ADD  one  ;
      STO  k    ;
      BR   loop ; goto loop
done: HLT       ; halt
```

Assembly Language

Language translation process

- Assemble the instructions to machine code.

Assembly Language

Language translation process

- Assemble the instructions to machine code.

```
loop: LDA  k
      SUB  ten
      BRZ  done
      LDA  i
      ADD  j
      STO  fib
      LDA  j
      STO  i
      LDA  fib
      STO  j
      LDA  i
      OUT
      LDA  k
      ADD  one
      STO  k
      BR   loop
done: HLT
```

Box	Code	Assembler
01	520	LDA k
02	222	SUB ten
03	717	BRZ done
04	519	LDA i
05	119	ADD j
06	321	STO fib
07	519	LDA j
08	319	STO i
09	521	LDA fib
10	319	STO j
11	519	LDA i
12	902	OUT
13	520	LDA k
14	123	ADD one
15	320	STO k
16	601	BR loop
17	000	HLT

Assembly Language

Language translation process

- Assemble the instructions to machine code.

```
loop: LDA  k
      SUB  ten
      BRZ  done
      LDA  i
      ADD  j
      STO  fib
      ...
      BR   loop
done: HLT
```

j is in box 18
i is in box 19
k is in box 20
fib is in box 21
ten is in box 22
one is in box 23

Box	Code	Assembler
01	520	LDA k
02	222	SUB ten
03	717	BRZ done
04	519	LDA i
05	119	ADD j
06	321	STO fib
07	519	LDA j
08	319	STO i
09	521	LDA fib
10	319	STO j
	519	LDA i
12	902	OUT
13	520	LDA k
14	123	ADD one
15	320	STO k
16	601	BR loop
17	000	HLT

Assembly Language

Summary

- High level languages are convenient to read and write for humans.
- Computers execute only binary machine code.
- Conversion between the two is required.
  - Compilers translate high level languages to machine code.
  - Assemblers translate assembly language into machine code.
- Use if/goto pseudo-code as an intermediate language between high level and assembler.

References

– Englander, I. (2009). The architecture of computer hardware and systems software: an information technology approach. Wiley.

Instruction set architecture ISA

CISC VS RISC

What is an Instruction Set?

- The complete collection of instructions that are understood by a CPU
- Machine Code
- Binary
- Usually represented by assembly codes

Instruction Set Architecture (ISA)

- Serves as an interface between software and hardware.
- Provides a mechanism by which the software tells the hardware what should be done.

High level language code : C, C++, Java, Fortran,

compiler

Assembly language code: architecture specific statements

assembler

Machine language code: architecture specific bit patterns

software

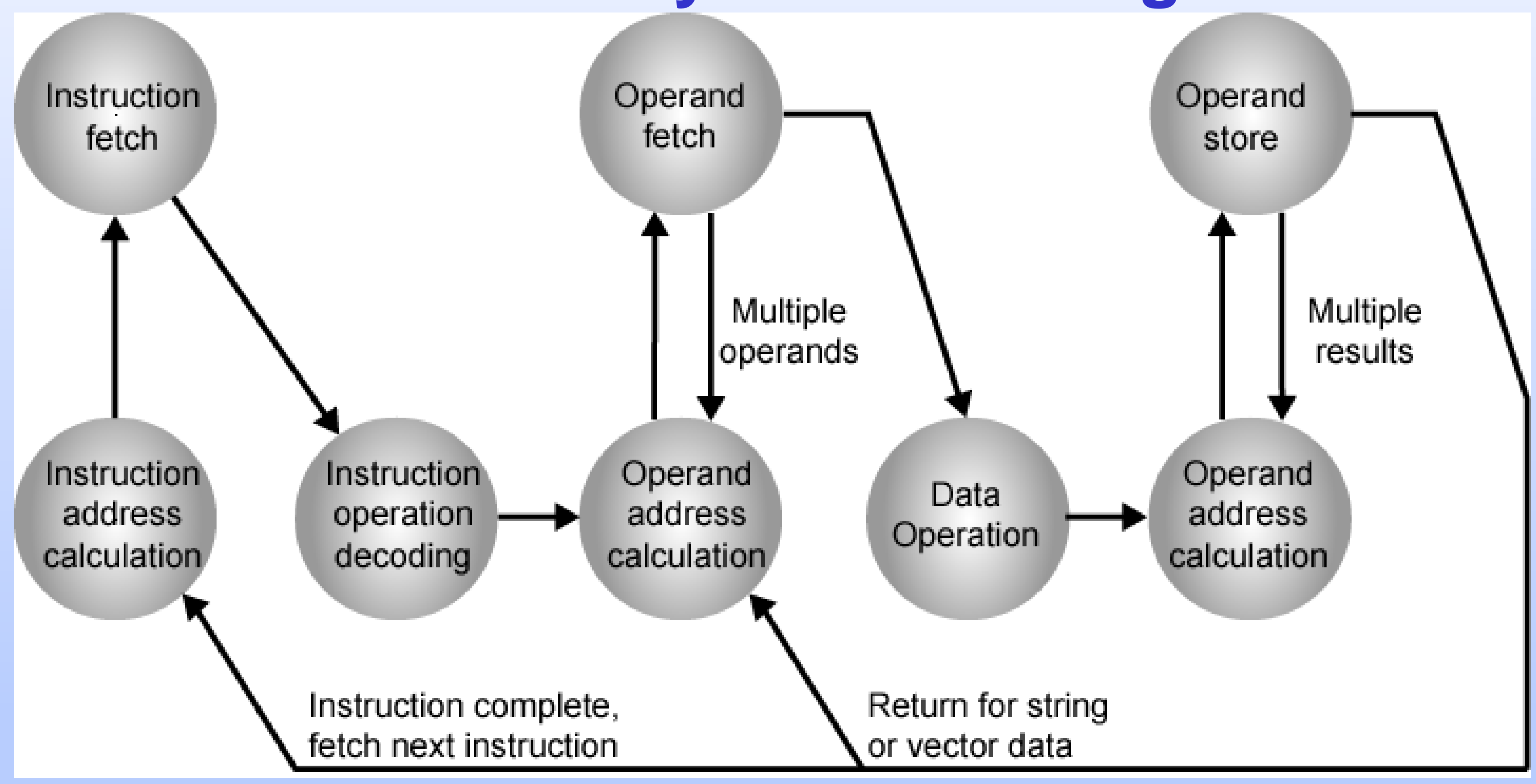
instruction set

hardware

Elements of an Instruction

- Operation code (Op code)
  - Do this
- Source Operand reference
  - To this
- Result Operand reference
  - Put the answer here
- Next Instruction Reference
  - When you have done that, do this...

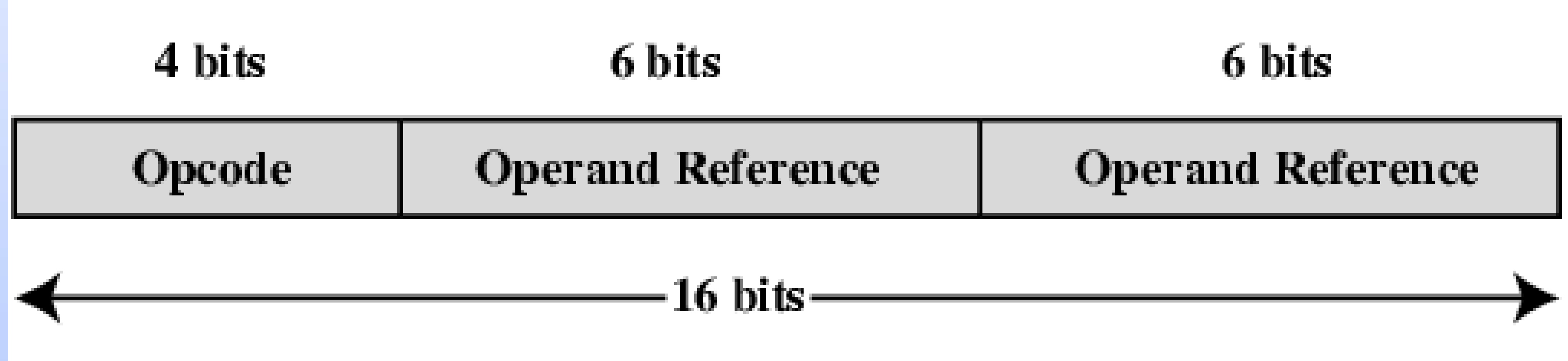
Instruction Cycle State Diagram



Instruction Representation

- In machine code each instruction has a unique bit pattern
- For human consumption (well, programmers anyway) a symbolic representation is used
  - e.g. ADD, SUB, LOAD
- Operands can also be represented in this way
  - ADD A,B

Simple Instruction Format

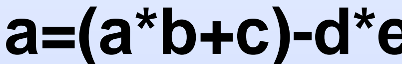


Instruction Types

- Data processing
- Data storage (main memory)
- Data movement (I/O)
- Program flow control

Number of Addresses (a)

- 3 addresses
  - Operand 1, Operand 2, Result
  - a = b + c;
  - May be a forth - next instruction (usually implicit)
  - Not common
  - Needs very long words to hold everything



Code	# Memory Refs
MPY T1, A, B	3
ADD T1, T1, C	3
MPY T2, D, E	3
SUB A, T1, T2	3

Number of Addresses (b)

- 2 addresses

a=(a*b+c)-d*e

  - One address doubles as operand and result
  - a = a + b
  - Reduces length of instruction
  - Requires some extra work
    - » Temporary storage to hold some results

Code	# Memory Refs
MOVE T1, A	2
MUL T1, B	3
ADD T1, C	3
MOVE T2, D	2
MUL T2, E	3
SUB T1, T2	3
MOVE A, T1	2

Number of Addresses (c)

- 1 address
  - Implicit second address
  - Usually a register (accumulator)
  - Common on early machines

a=(a*b+c)-d*e

Code	# Memory Refs
LOAD A	1
MUL B	1
ADD C	1
STORE T1	1
LOAD D	1
MUL E	1
STORE T2	1
LOAD T1	1
SUB T2	1
STORE A	1

Number of Addresses (d)

- 0 (zero) addresses
  - All addresses implicit
  - Uses a stack
  - e.g. push a
  - push b
  - add
  - pop c
  - c = a + b

$a=a*b+(c-(d*e))$

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

How Many Addresses

- More addresses
  - More complex (powerful?) instructions
  - More registers
    - Inter-register operations are quicker
  - Fewer instructions per program
- Fewer addresses
  - Less complex (powerful?) instructions
  - More instructions per program
  - Faster fetch/execution of instructions

Design Decisions (1)

- Operation repertoire
  - How many ops?
  - What can they do?
  - How complex are they?
- Data types
- Instruction formats
  - Length of op code field
  - Number of addresses

Design Decisions (2)

- Registers
  - Number of CPU registers available
  - Which operations can be performed on which registers?
- Addressing modes
- RISC v CISC

History of RISC/CISC

- 1950s IBM instituted a research program
- 1964 Release of System/360
- Mid-1970s improved measurement tools demonstrated on CISC
- 1975 801 project initiated at IBM's Watson Research Center
- 1979 32-bit RISC microprocessor (801) developed led by Joel Birnbaum
- 1984 MIPS developed at Stanford, as well as projects done at Berkeley
- 1988 RISC processors had taken over high-end of the workstation market
- Early 1990s IBM's POWER (*Performance Optimization With Enhanced RISC*) architecture introduced w/ the RISC System/6k
  - AIM (Apple, IBM, Motorola) alliance formed, resulting in PowerPC

What is CISC?

- CISC is an acronym for Complex Instruction Set Computer and are chips that are easy to program and which make efficient use of memory. Since the earliest machines were programmed in assembly language and memory was slow and expensive, the CISC philosophy made sense, and was commonly implemented in such large computers as the PDP-11 and the DECsystem 10 and 20 machines.
- Most common microprocessor designs such as the Intel 80x86 and Motorola 68K series followed the CISC philosophy.
- But recent changes in software and hardware technology have forced a re-examination of CISC and many modern CISC processors are hybrids, implementing many RISC principles.
- CISC was developed to make compiler development simpler. It shifts most of the burden of generating machine instructions to the processor. For example, instead of having to make a compiler write long machine instructions to calculate a square-root, a CISC processor would have a built-in ability to do this.

CISC Attributes

The design constraints that led to the development of CISC (small amounts of slow memory and fact that most early machines were programmed in assembly language) give CISC instructions sets some common characteristics:

- A 2-operand format, where instructions have a source and a destination. Register to register, register to memory, and memory to register commands. Multiple addressing modes for memory, including specialized modes for indexing through arrays
- Variable length instructions where the length often varies according to the addressing mode
- Instructions which require multiple clock cycles to execute.

E.g. Pentium is considered a modern CISC processor

Most CISC hardware architectures have several characteristics in common:

- **Complex instruction-decoding logic, driven by the need for a single instruction to support multiple addressing modes.**
- **A small number of *general purpose registers*. This is the direct result of having instructions which can operate directly on memory and the limited amount of chip space not dedicated to instruction decoding, execution, and microcode storage.**
- **Several *special purpose registers*. Many CTSC designs set aside special registers for the *stack pointer, interrupt handling,* and so on. This can simplify the hardware design somewhat, at the expense of making the instruction set more complex.**
- **A *'Condition code" register* which is set as a side-effect of most instructions. This register reflects whether the result of the last operation is less than, equal to, or greater than zero and records if certain error conditions occur.**

At the time of their initial development, CISC machines used available technologies to optimize computer performance.

- **Microprogramniing is as easy as assembly language to implement, and much less expensive than hardwiring a control unit.**
- **The ease of microcoding new instructions allowed designers to make CISC machines upwardly compatible: a new computer could run the same programs as earlier computers because the new computer would contain a superset of the instructions of the earlier computers.**
- **As each instruction became more capable, fewer instructions could be used to implement a given task. This made more efficient use of the relatively slow main memory.**
- **Because microprogram instruction sets can be written to match the constructs of high-level languages, the compiler does not have to be as complicated.**

CISC Disadvantages

Designers soon realised that the CISC philosophy had its own problems, including:

- Earlier generations of a processor family generally were contained as a subset in every new version - so instruction set & chip hardware become more complex with each generation of computers.
- So that as many instructions as possible could be stored in memory with the least possible wasted space, individual instructions could be of almost any length - this means that different instructions will take different amounts of clock time to execute, slowing down the overall performance of the machine.
- Many specialized instructions aren't used frequently enough to justify their existence -approximately 20% of the available instructions are used in a typical program.
- CISC instructions typically set the condition codes as a side effect of the instruction. Not only does setting the condition codes take time, but programmers have to remember to examine the condition code bits before a subsequent instruction changes them.

What is RISC?

- RISC?
  RISC, or *Reduced Instruction Set Computer*. is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures.
- History
  The first RISC projects came from IBM, Stanford, and UC-Berkeley in the late 70s and early 80s. The IBM 801, Stanford MIPS, and Berkeley RISC 1 and 2 were all designed with a similar philosophy which has become known as RISC. Certain design features have been characteristic of most RISC processors:
  - *one cycle execution time*: RISC processors have a CPI (clock per instruction) of one cycle. This is due to the optimization of each instruction on the CPU and a technique called PIPELINING
  - *pipelining*: a techique that allows for simultaneous execution of parts, or stages, of instructions to more efficiently process instructions;
  - *large number of registers*: the RISC design philosophy generally incorporates a larger number of registers to prevent in large amounts of interactions with memory

RISC Attributes

The main characteristics of CISC microprocessors are:

- Extensive instructions.
- Complex and efficient machine instructions.
- Microencoding of the machine instructions.
- Extensive addressing capabilities for memory operations.
- Relatively few registers.

In comparison, RISC processors are more or less the opposite of the above:

- Reduced instruction set.
- Less complex, simple instructions.
- Hardwired control unit and machine instructions.
- Few addressing schemes for memory operands with only two basic instructions, LOAD and
- STORE
- Many symmetric registers which are organised into a register file.

Pipelining

RISC Pipelines

A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

- fetch instructions from memory
- read registers and decode the instruction
- execute the instruction or calculate an address
- access an operand in data memory
- write the result into a register

RISC Disadvantages

- **There is still considerable controversy among experts about the ultimate value of RISC architectures. Its proponents argue that RISC machines are both cheaper and faster, and are therefore the machines of the future.**
- **However, by making the hardware simpler, RISC architectures put a greater burden on the software. Is this worth the trouble because conventional microprocessors are becoming increasingly fast and cheap anyway?**

CISC versus RISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

Summation

- As memory speed increased, and high-level languages displaced assembly language, the major reasons for CISC began to disappear, and computer designers began to look at ways computer performance could be optimized beyond just making faster hardware.
- One of their key realizations was that a sequence of simple instructions produces the same results as a sequence of complex instructions, but can be implemented with a simpler (and faster) hardware design. (Assuming that memory can keep up.) RISC (Reduced Instruction Set Computers) processors were the result.
- CISC and RISC implementations are becoming more and more alike. Many of today’s RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.
- To some extent, the argument is becoming moot because CISC and RISC implementations are becoming more and more alike. Many of today's RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.

CHAPTER 1: Computers and Systems

The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University

Typical Computer Ad

DREX

TECHNOLOGY CHOICES
TO BUILD YOUR BUSINESS

New! Vostre 400 Mini Tower

Only Drex delivers this combination of performance and expandibility in a system designed exclusively for Small Businesses—the Vostre 400.

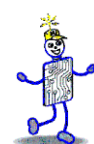
- Intex Core 2 Duo Processor
- Genuine Home Basic Operating System
- 2GB DDR2 SDRAM
- 160GB SATA Hard Drive
- 16x DVD+/-RW Drive
- 256MB PCI Express Graphics Card
- 1-Yr Limited Warranty, Next Business Day On-Site Service, and Hardware Warranty Support
- 20" Widescreen Flat Panel Display

ON SALE NOW!

NEW! Lower-priced upgrades:
Upgrade to 4GB Memory, 250GB Hard Drive, and 22" Widescreen Flat Panel Display for only $90!!

- Is the computer fast enough to run necessary programs?
- Is the computer cost-effective?
- Will it be obsolete in 6 months?

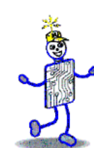
Copyright 2010 John Wiley & Sons, Inc.



Why Study Computer System Architecture?

- User
  - Understand system capabilities and limitations
  - Make informed decisions
  - Improve communications with information technology professionals
- Programmer
  - Create efficient application software for specific processing needs
- Systems Architect or Systems Analyst
  - Specify computer systems and architecture to meet application requirements
  - Make intelligent decisions about system strategy

Copyright 2010 John Wiley & Sons, Inc.

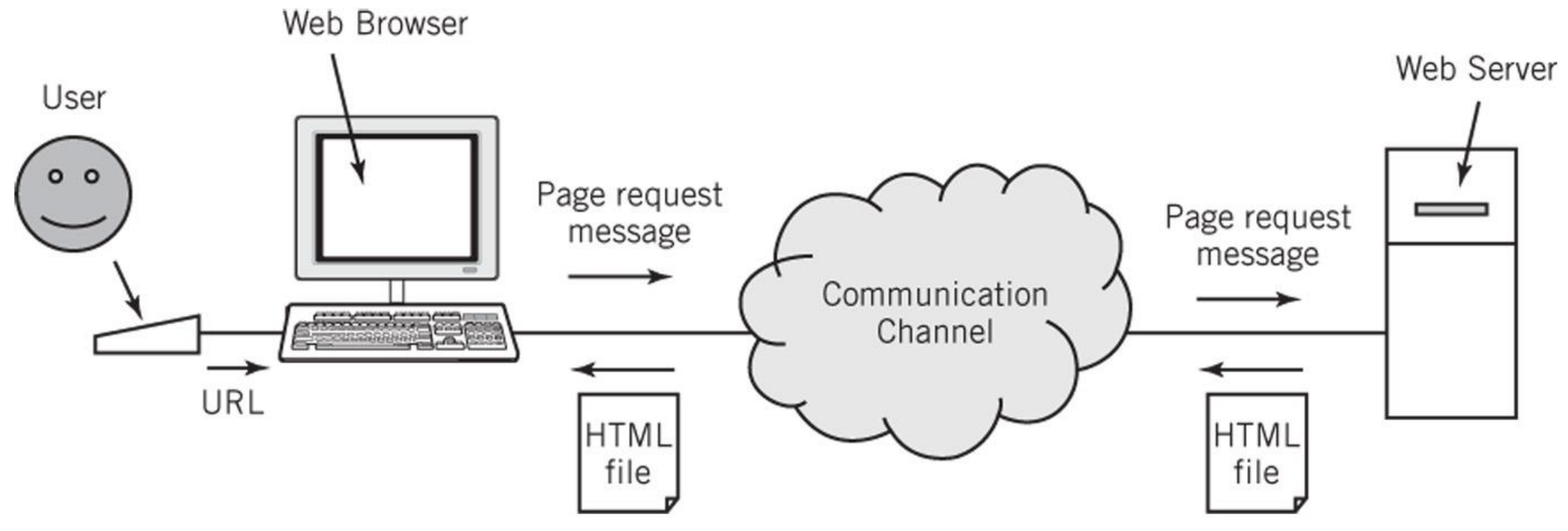


Why Study Computer System Architecture?

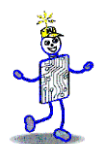
- System Administrator / Manager
  - Install, configure, maintain, and upgrade computer systems
  - Maximize system availability and efficiency
  - Optimize system performance
  - Ensure system security
- Web Services Designer
  - Optimize customer accessibility to Web services
  - Optimize web system configurations
  - Select appropriate data formats, page designs and scripting languages
  - Design efficient Web pages

Copyright 2010 John Wiley & Sons, Inc.

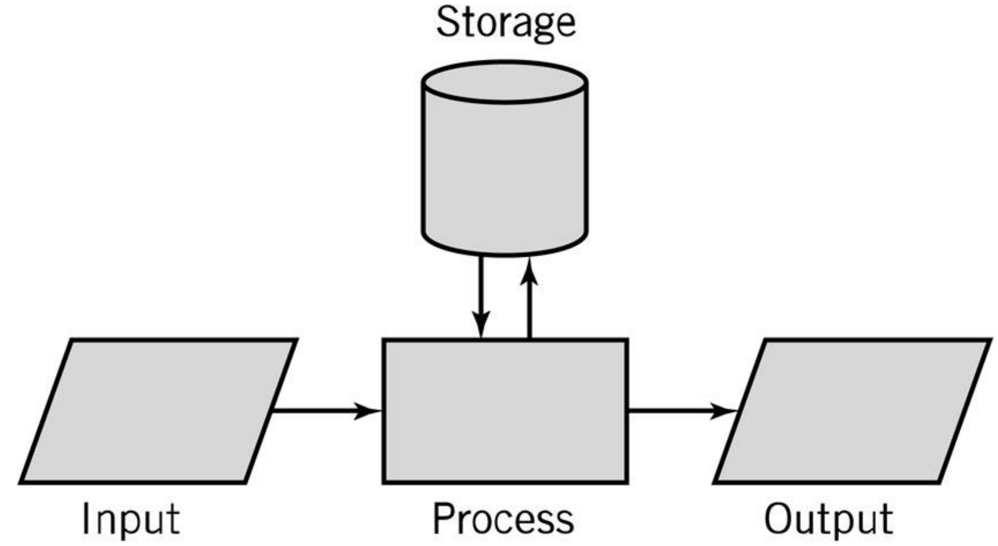
Web Browser Application Use



Copyright 2010 John Wiley & Sons, Inc.



Input-Process-Output Model (IPO)



- Input: keyboard, mouse, scanner, punch cards
- Processing: CPU executes the computer program
- Output: monitor, printer, fax machine
- Storage: hard drive, optical media, diskettes, magnetic tape

Copyright 2010 John Wiley & Sons, Inc.

Simplified IT Computer System Layout

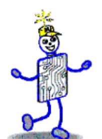
Intranet web server
Database server
Apps. server
Internet web server
Firewall
To Internet

Sales
Order entry
Service
Support
Web design
Printer

Order Fulfillment
Purchasing
Inventory
Warehousing
Printer
Shipping

Finance
Financial planning
Accounts receivable
Accounts payable
Printer
Credit
Accounting

Marketing
Research & planning
Printer
Sales
Advertising



Computer System Components

- *Hardware*
  - Processes data by executing instructions
  - Provides input and output
  - Control input, output and storage components
- *Software*
  - Applications and system software
  - Instructions tell hardware exactly what tasks to perform and in what order
- *Data*
  - Fundamental representation of facts and observations
- *Communications*
  - Sharing data and processing among different systems

Copyright 2010 John Wiley & Sons, Inc.

Hardware Component

- Input/Output devices
- Storage Devices
- CPU – Central Processing Unit
  - ALU: arithmetic/logic unit
  - CU: control unit
  - Interface unit
- Memory
  - Short-term storage for CPU calculations

Copyright 2010 John Wiley & Sons, Inc.

Typical Personal Computer System

Floppy disk

CD-R/RW or DVD

Communication interface

DATA

Hard disk

Keyboard

CPU

Memory

I/O Interface

Computer

Printer

Monitor

Tape

Mouse

Speaker

Copyright 2010 John Wiley & Sons, Inc.

CPU: Central Processing Unit

- ALU: arithmetic/logic unit
  - Performs arithmetic and Boolean logical calculations
- CU: control unit
  - Controls processing of instructions
  - Controls movement of data within the CPU
- Interface unit
  - Moves instructions and data between the CPU and other hardware components
  - *Bus*: bundle of wires that carry signals and power between different components

Copyright 2010 John Wiley & Sons, Inc.

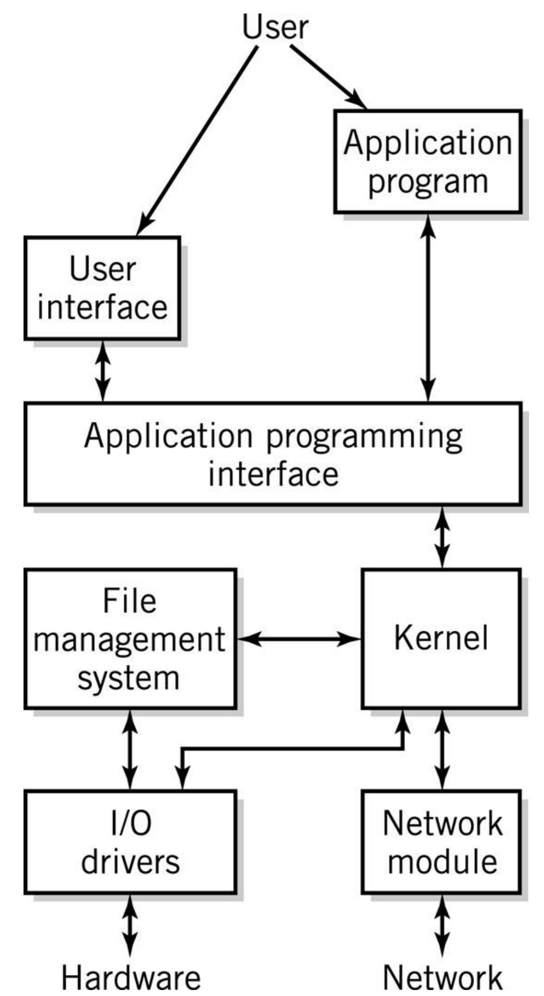
Memory

- Also known as *primary storage*, *working storage*, working storage, and *RAM (random access memory)*
- Consists of bits, each of which hold a value of either 0 or 1 (8 bits = 1 byte)
- Holds both instructions and data of a computer program (*stored program concept*)

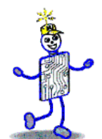
Copyright 2010 John Wiley & Sons, Inc.

Software Component

- Applications
- *Operating System*
  - API: application program interface
  - File management
  - I/O
  - Kernel
    - Memory management
    - Resource scheduling
    - Program communication
    - Security
  - Network Module



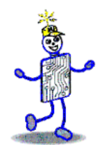
Copyright 2010 John Wiley & Sons, Inc.



Communications Component

- Hardware
  - Communication *channels*
    - Physical connections between computer systems
    - Examples: wire cable, phone lines, fiber optic cable, infrared light, radio waves
  - Interface hardware
    - Handles communication between the computer and the communication channel
    - *Modem* or *network interface card (NIC)*
- Software
  - Establish connections
  - Control flow of data
  - Directs data to the proper applications for use

Copyright 2010 John Wiley & Sons, Inc.



Computer Systems

All computer systems, no matter how complex, consists of the following:

- At least one CPU
- Memory to hold programs and data
- I/O devices
- Long-term storage

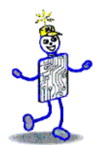
Copyright 2010 John Wiley & Sons, Inc.

Computer Systems Examples

IBM System z10 EC Mainframe

HP Laptop Computer

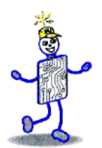
Copyright 2010 John Wiley & Sons, Inc.



Virtualization

- Virtual (American Heritage Dictionary
  - Existing or result in essence or effect though not in actual fact, form or name
  - Created, simulated, or carried on by means of a computer or computer network
- Computer systems examples
  - Virtual memory
  - Virtual networks
  - Java Virtual Machine

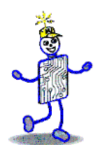
Copyright 2010 John Wiley & Sons, Inc.



Protocols

- Common ground rules of communication between computers, I/O devices, and many software programs
- Examples
  - HTTP: between Web servers and Web browsers
  - TCP/IP: between computers on the Internet and local area networks
  - SATA: between storage devices and computers
  - XML,RSS, SIP: new protocols

Copyright 2010 John Wiley & Sons, Inc.



Standards

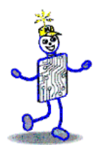
- Created to ensure universal compatibility of data formats and protocols
- May be created by committee or may become a de facto standard through popular use
- Examples:
  - Computer languages: Java, SQL, C, JavaScript
  - Display standards: Postscript, MPEG-2, JPEG, GIF
  - Character set standards: ASCII, Unicode, EBCDIC
  - Multimedia standards: MPEG-2, MPEG-4, DivX, MP3

Copyright 2010 John Wiley & Sons, Inc.

Textbook Overview

- Web site: http://www.wiley.com/college/englander
- Part 1 (Chapters 1-2)
  - Overview of computer systems
- Part 2 (Chapters 3-5)
  - Number systems and data formats
- Part 3 (Chapters 6-11)
  - Computer architecture and hardware operation
- Part 4 (Chapters 12-14)
  - Networks and data communications
- Part 5 (Chapters 15-18)
  - Software component – operating systems
- Part 6 (Supplementary Chapters S1-S4)
  - Digital logic, systems examples, instruction addressing modes, programming tools

Copyright 2010 John Wiley & Sons, Inc.



Early History

- 1642: Blaise Pascal invents a calculating machine
- 1801: Joseph Marie Jacquard invents a loom that uses punch cards
- 1800's:
  - Charles Babbage attempts to build an analytical engine (mechanical computer)
  - Augusta Ada Byron develops many of the fundamental concepts of programming
  - George Boole invents Boolean logic.

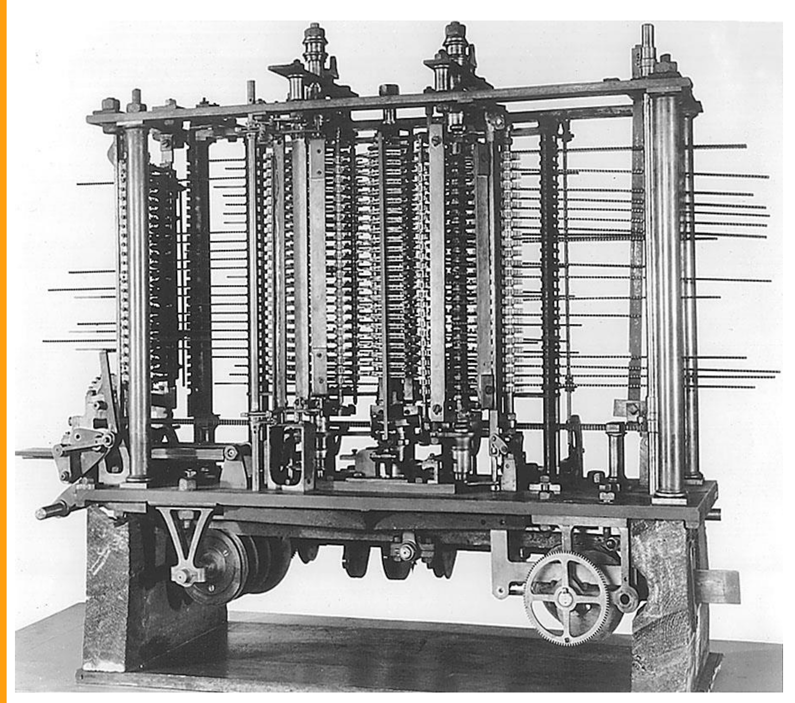
Copyright 2010 John Wiley & Sons, Inc.

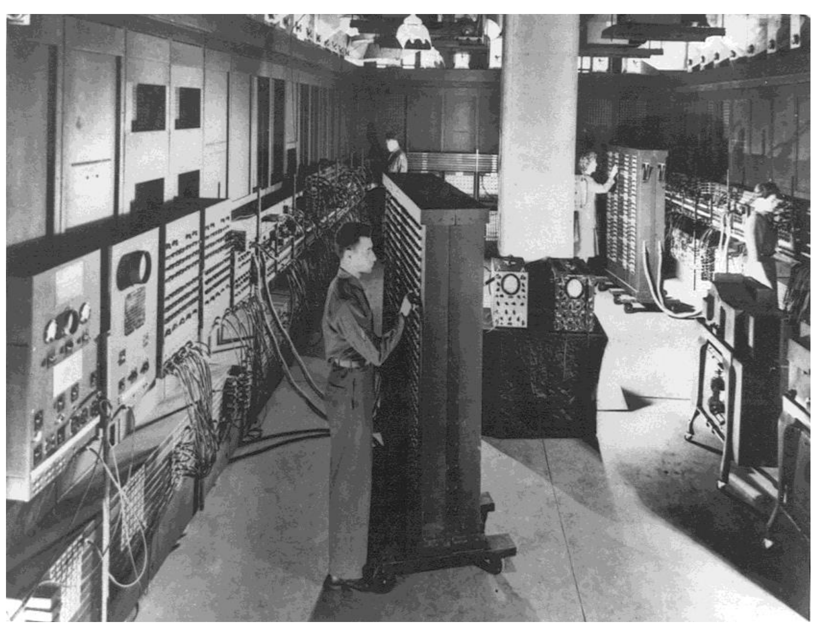
Modern Computer Development

- 1937: Mark I is built (Aiken, Harvard University, IBM).
  - First electronic computer using relays.
- 1939: ABC is built
  - First fully electronic digital computer. Used vacuum tubes.
- 1943-46: ENIAC (Mauchly, Eckert, University of Pennsylvania).
  - First general purpose digital computer.
- 1945: Von Neumann architecture proposed.
  - Still the standard for present day computers.
- 1947: Creation of transistor
  - (Bardeen, Shockley, Brattain, Bell Labs).
- 1951-2: EDVAC and IAS

Copyright 2010 John Wiley & Sons, Inc.

Early Computers

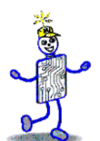




Babbage's Analytical Engine

ENIAC

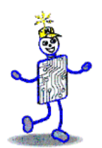
Copyright 2010 John Wiley & Sons, Inc.



System Software History

- Early computers had no operating systems and were single user systems
  - Programs were entered using switches for each bit or by plugging wires into a panel
- 1953-54: First operating system was built by General Motors Research Laboratories for their IBM 701 computer
- Other early systems
  - FORTRAN Monitor System (FMS)
  - IBSYS
  - Share Operating System (SOS)

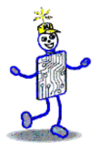
Copyright 2010 John Wiley & Sons, Inc.



Operating System Development

- 1963: Master Control Program (MCP) by Burroughs. Included many modern OS features.
- 1964: OS/360 by IBM. Included batch processing of programs.
- 1962: MIT Project MAC created a time-sharing OS called CTSS. Shortly afterwards, MIT, Bell Labs, and GE developed Multics (Multiplexed Information and Computing Services).

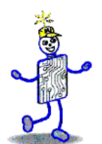
Copyright 2010 John Wiley & Sons, Inc.



UNIX

- After Bell Labs withdrew from the Multics project, Ken Thompson developed a personal operating system called UNIX using assembly language.
- Dennis Ritchie developed the programming language C which was used to rewrite much of UNIX in a high-level language.
- UNIX introduced
  - A hierarchical file system
  - The shell concept
  - Document production and formatting
  - Tools for networked and distributed processing

Copyright 2010 John Wiley & Sons, Inc.



Graphical User Interfaces

- 1960s: Doug Englebart (Stanford Research Institute)
  - Invented windows and a mouse interface
- 1970s: Xerox PARC
  - Creates a practical windowing system for the Dynabook project
- 1980s: Steve Jobs (Apple)
  - Developed the Apple Lisa and MacIntosh

Copyright 2010 John Wiley & Sons, Inc.

IBM PC

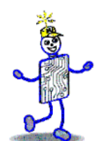
- 1982: Stand-alone, single user computer
- PC-DOS, MS-DOS (disk operating system)
- Later versions of DOS added
  - Hierarchical directory file storage
  - File redirection
  - Better memory management
- Windowing systems
  - Windows 2.0, Windows 3.1, Windows 95
  - Windows NT, Windows XP, Windows Vista
  - Windows 7

Copyright 2010 John Wiley & Sons, Inc.

Communications

- 1960s and 1970s: users communicated on multiterminal computer systems using talk and email facilities
- 1971: Ray Tomlinson creates the standard username@hostname email standard
- Modems permitted users to login to office systems, electronic bulletin board systems, Compuserve, AOL, and Prodigy
- 1969: ARPANET begun
- 1985: First TCP-IP wide area network
- 1991: Tim Berners Lee develops the concepts that become the World Wide Web
- 1993: Max Andreessen develops Mosaic, the first graphical browser

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."

CHAPTER 2:
Introduction to Systems Concepts and Systems Architecture

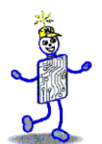
The Architecture of Computer Hardware, Systems Software & Networking:
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

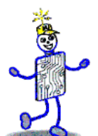
PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University



What is a system?

- What do the following systems have in common?
  1. Plumbing system
  2. Solar system
  3. Home network system
  4. Inventory control system

Copyright 2010 John Wiley & Sons, Inc.



Plumbing System

Water supply system

⅜" to 1" supply pipe

Air chamber

Shutoff

Shutoff

Shutoff

Shutoff

Relief valve

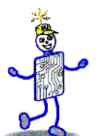
Drain

Main shutoff

Water meter

Main service pipe from water supply

Copyright 2010 John Wiley & Sons, Inc.



Solar System

Mercury
Earth
Venus
Mars
Jupiter
Saturn
Neptune
Uranus
Sun

Copyright 2010 John Wiley & Sons, Inc.

Home Network System

Phone line or cable

DSL or cable modem

Wireless router

Network-Attached Storage (NAS)

Network-Ready Printer

Inventory Control System

Sales order

1
Check order against inventory

Inventory

2a
Not Available

2b
Available; deduct from inventory

3
Check against inventory reorder point

To sales to notify customer

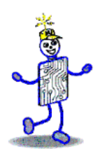
To purchasing to determine availability and shipping date

To warehouse for order fulfillment and shipping

[Stock < minimum]

To purchasing for reorder

Copyright 2010 John Wiley & Sons, Inc.



Definition of a System

- "A systems is a collection of components linked together and organized in such a way as to be recognizable as a single unit."
- Linked components of a system also define the boundary for the system
- The environment is anything outside of the system

Copyright 2010 John Wiley & Sons, Inc.

General Representation of a System

Environment

Environment

Interface

Boundary

Interface

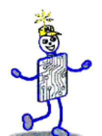
Links

Environment

Components

Environment

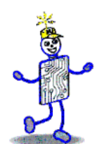
Copyright 2010 John Wiley & Sons, Inc.



System Decomposition

- Components
  - May be irreducible or
  - May be subsystems
- Decomposition
  - The division of a system into its components and linkages
  - Hierarchical

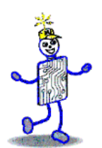
Copyright 2010 John Wiley & Sons, Inc.



System Architecture

"The fundamental properties, and the patterns of relationships, connections, constraints, and linkages among the components and between the system and its environment are known collectively as the architecture of the system"

Copyright 2010 John Wiley & Sons, Inc.



Abstractions of Systems

- How are the following two abstractions of a business system different from one another?
- How are these abstractions different from the real business system?

Copyright 2010 John Wiley & Sons, Inc.

Business Organization Chart

- Corporate exec. management
  - Marketing and sales
    - Sales
    - Advertising
    - Planning
    - Order Fulfillment
  - IT
    - System planning & development
    - System Administration
    - User support
  - Human Resources
    - Employment
    - Organizational development
    - Contracts
  - Finance
    - Accounting
    - Financial planning
    - Purchasing
    - Auditing & control

Copyright 2010 John Wiley & Sons, Inc.

Business Application Architecture

Marketing Information System

Executive Information System

Customers

Accounts Receivable

Order Entry System

Orders

Order fulfillment

Financial Information System

Products

Purchasing

Suppliers

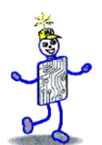
Accounts payable

Copyright 2010 John Wiley & Sons, Inc.

IT System Architectures

- Distributed processing systems
  - Client-Server Computing
    - 2-tier architecture
    - 3-tier architecture
    - N-tier architecture
    - Web-Based Computing
  - Peer-to-Peer Computing

Copyright 2010 John Wiley & Sons, Inc.



Client-Server Computing

- A program on a client computer requests services from a program on a server computer
- Examples:
  - Email services, file services, print services, directory services, Web services, database services, application services, remote access services

Copyright 2010 John Wiley & Sons, Inc.

Basic Client-Server Architecture

Client

Server

Request

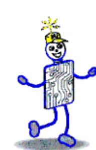
Request

Service response

Service response

Communication channel

Copyright 2010 John Wiley & Sons, Inc.



Advantages of Client-Server Architecture

- Centralization of services permits
  - easier administration of services by IT professionals
  - easier availability and location by users
  - consistency of resources, such as files and data, can be managed and assured
  - more efficient and cost-effective hardware procurement through purchasing a small number of very powerful computers

Copyright 2010 John Wiley & Sons, Inc.

Clients and Servers on a Network

Servers

Web server

Database server

File server

E-mail server

S2

Application server & print server

Network

Clients

Multi-tier Architectures

- Two-tier architecture
  - Two computers are involved in a service.
  - Example: Web-browser and Web server model used in intranets and on the Internet
- Three-tier architecture
  - Three computers are involved in a service
  - Example: client computer, Web server, database server
- N-tier architecture

Copyright 2010 John Wiley & Sons, Inc.

Three-tier Architecture

Web server

Mail server

HTTP Request

HTTP response

CGI* Request

CGI response (HTML)

R'cvd mail

Sent mail

SMTP* to another mail server

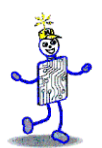
*SMTP: Simple Mail Transfer Protocol
*CGI: Common Gateway Interface

Copyright 2010 John Wiley & Sons, Inc.

Peer-to-Peer Computing

- Computers on a network are treated as equals
- Each computer can share resources with the other computers on the network
- Disadvantages
  - Difficult to establish centralized control of services
  - Difficult to locate services
  - Difficult to synchronize versions of files or software
  - Difficult to secure network from unauthorized access and from viruses
- Advantages
  - Sharing files between personal computers
  - Internet file sharing

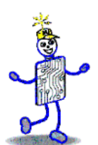
Copyright 2010 John Wiley & Sons, Inc.



Hybrid Model of Computing

- Client-server technology used to locate systems and files
- Then systems can participate in peer-to-peer transactions
- Examples
  - Instant messaging
  - Skype
  - Napster

Copyright 2010 John Wiley & Sons, Inc.



Google: System Architecture

- Provide powerful, fast search capability for material on the Internet
- Derive income from advertising that is targeted to each user based on their searches
- Basic requirements
  - Capable of responding to millions of simultaneous requests from all over the world
  - Perform a web crawl of the Internet retrieve and organize data
  - Establish ranking of results with appropriately targeted advertising
  - High reliability of the system
  - System is easily scalable and cost effective

Copyright 2010 John Wiley & Sons, Inc.

Google Data Center Search Application Architecture

Internet

Interface to internet

Spell Checker

Web Servers

Ad Checker

Index databases

Index Servers

Web Page Document Servers

Page databases

Copyright 2010 John Wiley & Sons, Inc.

Simplified Google System Hardware Architecture

To Internet

Network switch

...
Up to 64 lines

Network switch

Network switch

Network switch

...

...

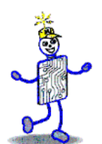
...

Up to 80 PCs

Up to 80 PCs

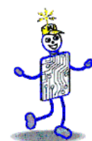
Up to 80 PCs

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 3: Number Systems

The Architecture of Computer Hardware and Systems Software & Networking:

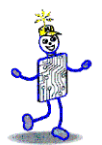
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

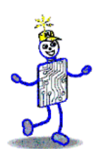
PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University



Why Binary?

- Early computer design was decimal
  - Mark I and ENIAC
- John von Neumann proposed binary data processing (1945)
  - Simplified computer design
  - Used for both instructions and data
- Natural relationship between on/off switches and calculation using Boolean logic

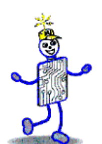
On	Off
True	False
Yes	No
1	0



Counting and Arithmetic

- Decimal or base 10 number system
  - Origin: counting on the fingers
  - "Digit" from the Latin word *digitus* meaning "finger"
- *Base*: the number of different digits including zero in the number system
  - Example: Base 10 has 10 digits, 0 through 9
- *Binary* or *base 2*
- *Bit* (binary digit): 2 digits, 0 and 1
- *Octal* or *base 8:* 8 digits, 0 through 7
- *Hexadecimal or base 16:* 16 digits, 0 through F
  - Examples: $10_{10} = A_{16}$; $11_{10} = B_{16}$

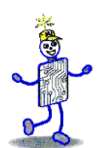
Copyright 2010 John Wiley & Sons, Inc.



Keeping Track of the Bits

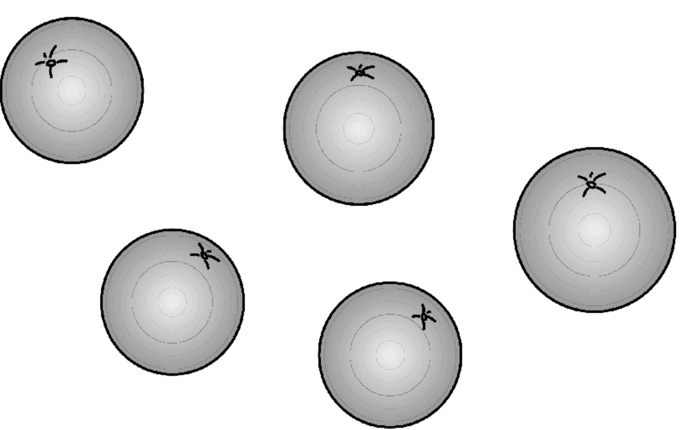
- Bits commonly stored and manipulated in groups
  - 8 bits = 1 *byte*
  - 4 bytes = 1 word (in many systems)
- Number of bits used in calculations
  - Affects accuracy of results
  - Limits size of numbers manipulated by the computer

Copyright 2010 John Wiley & Sons, Inc.

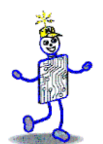


Numbers: Physical Representation

- Different numerals, same number of oranges
  - Cave dweller: IIIII
  - Roman: V
  - Arabic: 5
- Different bases, same number of oranges
  - 5_{10}
  - 101_{2}
  - 12_{3}



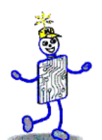
Copyright 2010 John Wiley & Sons, Inc.



Number System

- Roman: position *independent*
- Modern: based on positional notation (place value)
  - Decimal system: system of **positional** notation based on powers of 10.
  - Binary system: system of **positional** notation based powers of 2
  - Octal system: system of **positional** notation based on powers of 8
  - Hexadecimal system: system of **positional** notation based powers of 16

Copyright 2010 John Wiley & Sons, Inc.



Positional Notation: Base 10

$527 = 5 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

100's place

10's place

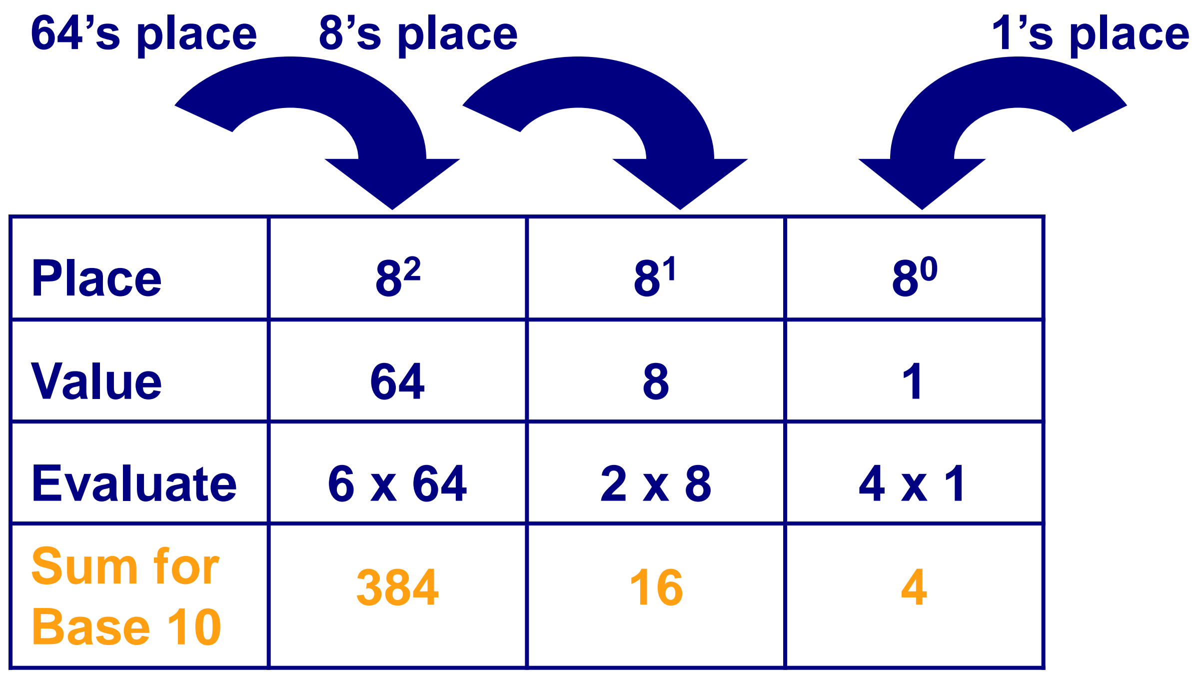
1's place

Place	10^2	10^1	10^0
Value	100	10	1
Evaluate	5 x 100	2 x 10	7 x1
Sum	500	20	7

Copyright 2010 John Wiley & Sons, Inc.

Positional Notation: Octal

$624_8 = 404_{10}$



Place	8^2	8^1	8^0
Value	64	8	1
Evaluate	6 x 64	2 x 8	4 x 1
Sum for Base 10	384	16	4

Copyright 2010 John Wiley & Sons, Inc.

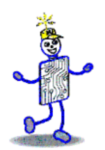
Positional Notation: Hexadecimal

$6{,}704_{16} = 26{,}372_{10}$

4,096's place   256's place   16's place   1's place

Place	16^3	16^2	16^1	16^0
Value	4,096	256	16	1
Evaluate	6 x 4,096	7 x 256	0 x 16	4 x 1
Sum for Base 10	24,576	1,792	0	4

Copyright 2010 John Wiley & Sons, Inc.

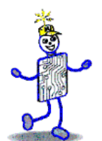


Positional Notation: Binary

$1101\ 0110_2 = 214_{10}$

Place	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	128	64	32	16	8	4	2	1
Evaluate	1 x 128	1 x 64	0 x 32	1 x16	0 x 8	1 x 4	1 x 2	0 x 1
Sum for Base 10	128	64	0	16	0	4	2	0

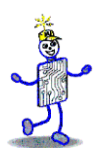
Copyright 2010 John Wiley & Sons, Inc.



Range of Possible Numbers

- $R = B^K$ where
  - R = range
  - B = base
  - K = number of digits
- Example #1: Base 10, 2 digits
  - $R = 10^2 =$ 100 different numbers (0…99)
- Example #2: Base 2, 16 digits
  - $R = 2^{16} =$ 65,536 or 64K
  - 16-bit PC can store 65,536 different number values

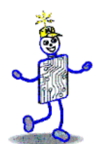
Copyright 2010 John Wiley & Sons, Inc.



Decimal Range for Bit Widths

Bits	Digits	Range
1	0+	2 (0 and 1)
4	1+	16 (0 to 15)
8	2+	256
10	3	1,024 (1K)
16	4+	65,536 (64K)
20	6	1,048,576 (1M)
32	9+	4,294,967,296 (4G)
64	19+	Approx. 1.6×10^{19}
128	38+	Approx. 2.6×10^{38}

Copyright 2010 John Wiley & Sons, Inc.



Base or Radix

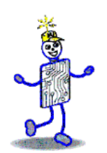
- Base:
  - The number of different symbols required to represent any given number
- The *larger* the base, the *more* numerals are required
  - Base 10: 0,1, 2,3,4,5,6,7,8,9
  - Base 2: 0,1
  - Base 8: 0,1,2, 3,4,5,6,7
  - Base 16: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Copyright 2010 John Wiley & Sons, Inc.

Number of Symbols vs. Number of Digits

- For a given number, the *larger* the base
  - the *more* symbols required
  - but the *fewer* digits needed
- Example #1:
  - 65_{16} 101_{10} 145_{8} $110\ 0101_{2}$
- Example #2:
  - $11C_{16}$ 284_{10} 434_{8} $1\ 0001\ 1100_{2}$

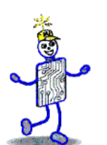
Copyright 2010 John Wiley & Sons, Inc.



Counting in Base 2

Binary Number	Equivalent				Decimal Number
	8's (2^3)	4's (2^2)	2's (2^1)	1's (2^0)	
0				0×2^0	0
1				1×2^0	1
10			1×2^1	0×2^0	2
11			1×2^1	1×2^0	3
100		1×2^2			4
101		1×2^2		1×2^0	5
110		1×2^2	1×2^1		6
111		1×2^2	1×2^1	1×2^0	7
1000	1×2^3				8
1001	1×2^3			1×2^0	9
1010	1×2^3		1×2^1		10

Copyright 2010 John Wiley & Sons, Inc.



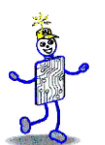
Base 10 Addition Table

$3_{10} + 6_{10} = 9_{10}$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13

etc

Copyright 2010 John Wiley & Sons, Inc.



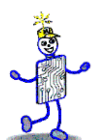
Base 8 Addition Table

$3_8 + 6_8 = 11_8$

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

(no 8 or 9, of course)

Copyright 2010 John Wiley & Sons, Inc.



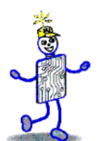
Base 10 Multiplication Table

$3_{10} \times 6_{10} = 18_{10}$

x	0	1	2	3	4	5	6	7	8	9
0				0						
1		1	2	3	4	5	6	7	8	9
2		2	4	6	8	10	12	14	16	18
3		3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5		5	10	15	20	25	30	35	40	45
6		6	12	18	24	30	36	42	48	54
7		7	14	21	28	35	42	49	56	63

etc.

Copyright 2010 John Wiley & Sons, Inc.



Base 8 Multiplication Table

$3_8 \times 6_8 = 22_8$

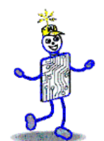
x	0	1	2	3	4	5	6	7
0				0				
1		1	2	3	4	5	6	7
2		2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4		4	10	14	20	24	30	34
5		5	12	17	24	31	36	43
6		6	14	22	30	36	44	52
7		7	16	25	34	43	52	61

Copyright 2010 John Wiley & Sons, Inc.

Addition

Base	Problem	Largest Single Digit
Decimal	6 +3	9
Octal	6 +1	7
Hexadecimal	6 +9	F
Binary	1 +0	1

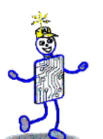
Copyright 2010 John Wiley & Sons, Inc.



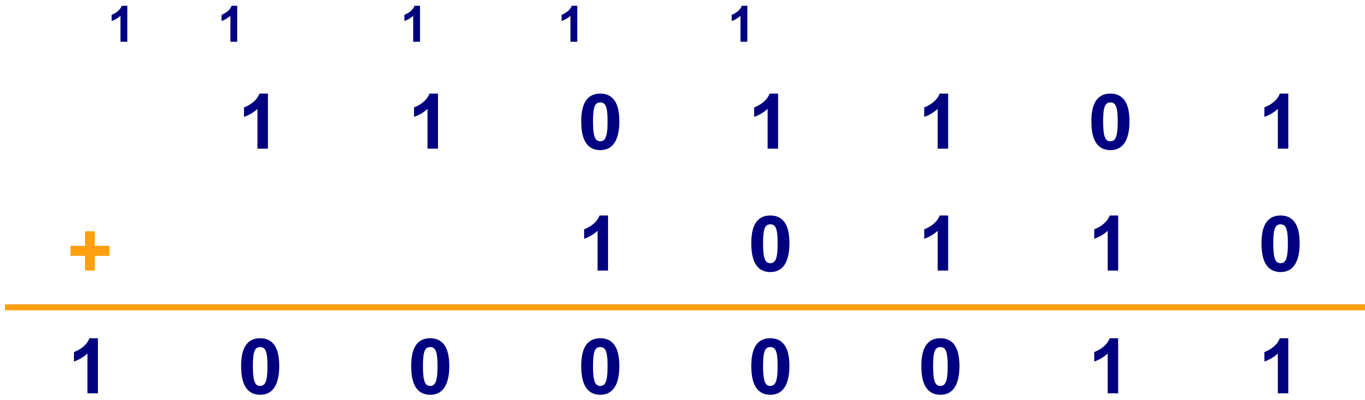
Addition

Base	Problem	Carry	Answer
Decimal	6 +4	Carry the 10	10
Octal	6 +2	Carry the 8	10
Hexadecimal	6 +A	Carry the 16	10
Binary	1 +1	Carry the 2	10

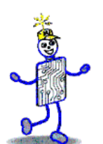
Copyright 2010 John Wiley & Sons, Inc.



Binary Arithmetic



Copyright 2010 John Wiley & Sons, Inc.



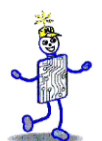
Binary Arithmetic

- Addition
  - Boolean using XOR and AND
- Multiplication
  - AND
  - Shift
- Division

+	0	1
0	0	1
1	1	10

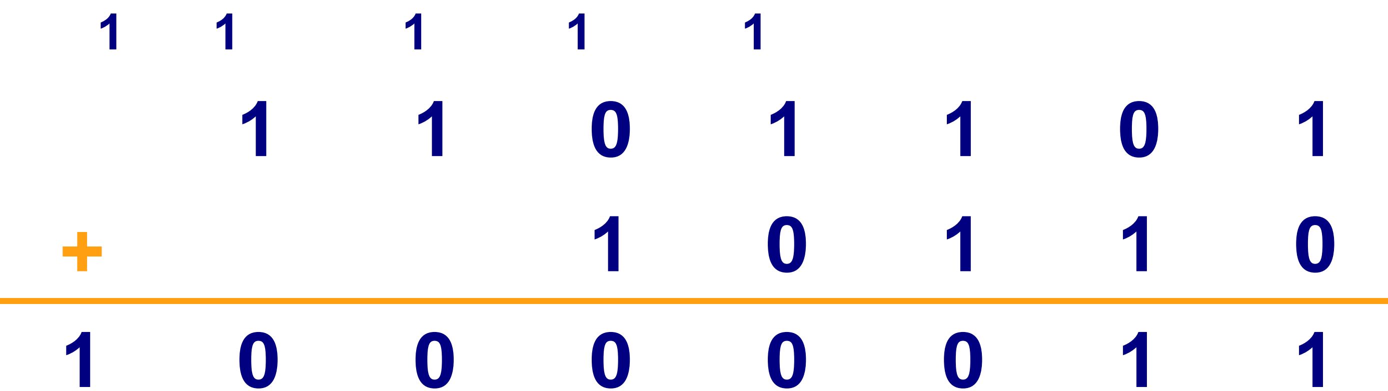
x	0	1
0	0	0
1	0	1

Copyright 2010 John Wiley & Sons, Inc.

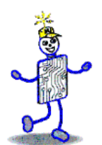


Binary Arithmetic: Boolean Logic

- Boolean logic without performing arithmetic
  - *EXCLUSIVE-OR*
    - Output is “1” only if either input, but ***not*** both inputs, is a “1”
  - *AND (carry bit)*
    - Output is “1” if and only both inputs are a “1”



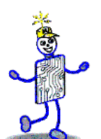
Copyright 2010 John Wiley & Sons, Inc.



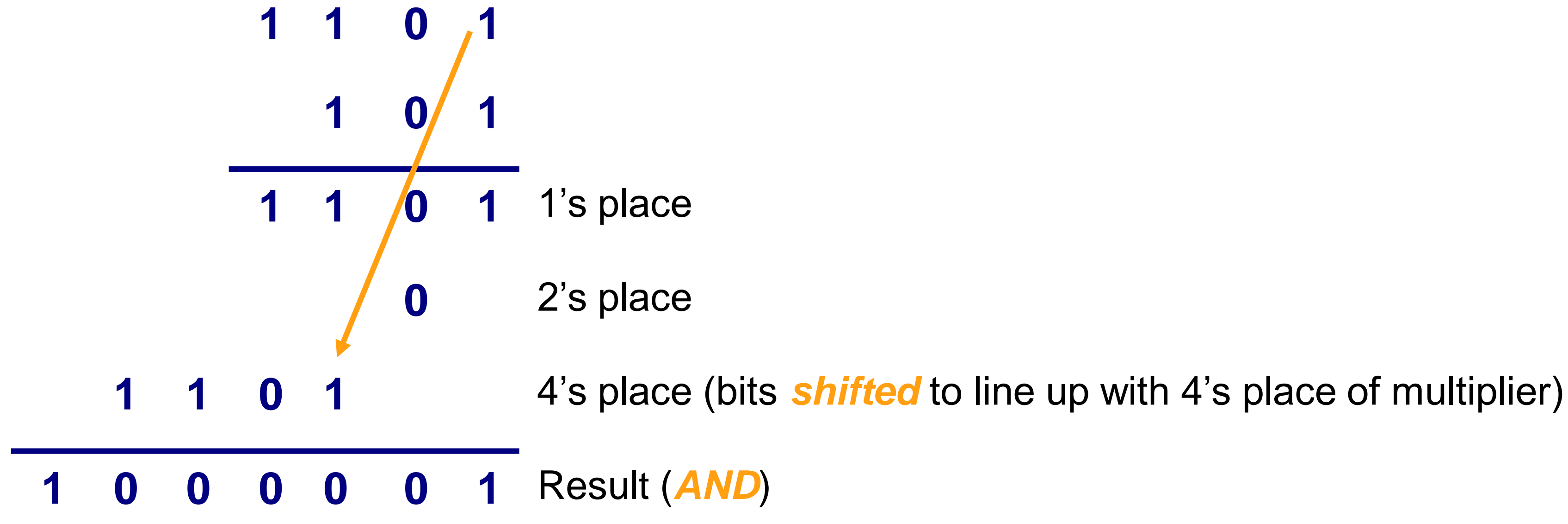
Binary Multiplication

- Boolean logic without performing arithmetic
  - *AND (carry bit)*
    - Output is “1” if and only both inputs are a “1”
  - *Shift*
    - Shifting a number in any base ***left*** one digit ***multiplies*** its value by the base
    - Shifting a number in any base ***right*** one digit ***divides*** its value by the base
    - Examples:
      - 10_{10} shift ***left*** = 100_{10}
      - 10_{10} shift ***right*** = 1_{10}
      - 10_2 shift ***left*** = 100_2
      - 10_2 shift ***right*** = 1_2

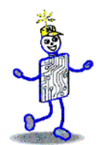
Copyright 2010 John Wiley & Sons, Inc.



Binary Multiplication



Copyright 2010 John Wiley & Sons, Inc.

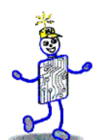


Converting from Base 10

- Powers Table

Base \ Power	8	7	6	5	4	3	2	1	0
2	256	128	64	32	16	8	4	2	1
8				32,768	4,096	512	64	8	1
16					65,536	4,096	256	16	1

Copyright 2010 John Wiley & Sons, Inc.

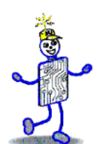


From Base 10 to Base 2

$42_{10} = 101010_2$

Power / Base	6	5	4	3	2	1	0
2	64	32	16	8	4	2	1
		1	0	1	0	1	0
Integer		42/32 = 1	10/16 = 0	10/8 = 1	2/4 = 0	2/2 = 1	0/1 = 0
Remainder		10	10	2	2	0	0

Copyright 2010 John Wiley & Sons, Inc.



From Base 10 to Base 2

Base 10 42

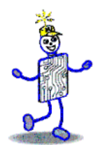
Remainder

Quotient

2)	42	(**0**	Least significant bit
2)	21	(**1**	
2)	10	(**0**	
2)	5	(**1**	
2)	2	(**0**	
2)	**1**		Most significant bit

Base 2 **101010**

Copyright 2010 John Wiley & Sons, Inc.

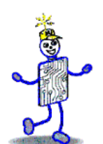


From Base 10 to Base 16

$5{,}735_{10} = 1667_{16}$

Power / Base	4	3	2	1	0
16	65,536	4,096	256	16	1
		1	**6**	**6**	**7**
Integer		5,735 /4,096 = 1	1,639 / 256 = 6	103 /16 = 6	7
Remainder		5,735 - 4,096 = 1,639	1,639 –1,536 = 103	103 – 96 = 7	

Copyright 2010 John Wiley & Sons, Inc.



From Base 10 to Base 16

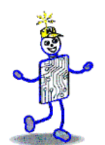
Base 10 8,039

Remainder

Quotient

16)	8,039	(7	Least significant bit
16)	502	(6	
16)	31	(15	
16)	1	(1	Most significant bit
16)	0		

Base 16 1F67



From Base 8 to Base 10

$7263_8 = 3,763_{10}$

Power	8^3	8^2	8^1	8^0
	512	64	8	1
	x 7	x 2	x 6	x 3
Sum for Base 10	3,584	128	48	3

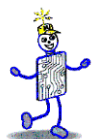
Copyright 2010 John Wiley & Sons, Inc.

From Base 8 to Base 10

$7263_8 = 3{,}763_{10}$

$$
\begin{array}{r}
\mathbf{7} \\
\underline{\times\ 8} \\
56 \quad + \mathbf{2} = \quad 58 \\
\underline{\times\ 8} \\
464 \quad + \mathbf{6} = \quad 470 \\
\underline{\times\ 8} \\
3760 \quad + \mathbf{3} = \quad 3{,}763
\end{array}
$$

Copyright 2010 John Wiley & Sons, Inc.



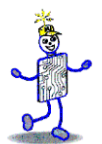
From Base 16 to Base 2

- The nibble approach
  - Hex easier to read and write than binary

Base 16	1	F	6	7
Base 2	0001	1111	0110	0111

- Why hexadecimal?
  - Modern computer operating systems and networks present variety of troubleshooting data in hex format

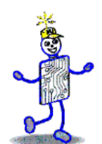
Copyright 2010 John Wiley & Sons, Inc.



Fractions

- Number point or radix point
  - Decimal point in base 10
  - Binary point in base 2
- No exact relationship between fractional numbers in different number bases
  - Exact conversion may be impossible

Copyright 2010 John Wiley & Sons, Inc.



Decimal Fractions

- Move the number point one place to the right
  - Effect: multiplies the number by the base number
  - Example: 139.0_{10} → 1390_{10}
- Move the number point one place to the left
  - Effect: divides the number by the base number
  - Example: 139.0_{10} → 13.9_{10}

Copyright 2010 John Wiley & Sons, Inc.

Fractions: Base 10 and Base 2

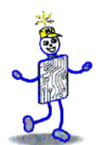
$.2589_{10}$

Place	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Value	1/10	1/100	1/1000	1/10000
Evaluate	2 x 1/10	5 x 1/100	8 x 1/1000	9 x1/1000
Sum	.2	.05	.008	.0009

$.101011_2 = 0.671875_{10}$

Place	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
Value	1/2	1/4	1/8	1/16	1/32	1/64
Evaluate	1 x 1/2	0 x 1/4	1x 1/8	0 x 1/16	1 x 1/32	1 x 1/64
Sum	.5		0.125		0.03125	0.015625

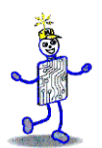
Copyright 2010 John Wiley & Sons, Inc.



Fractions: Base 10 and Base 2

- No general relationship between fractions of types $1/10^k$ and $1/2^k$
  - Therefore a number representable in base 10 may not be representable in base 2
  - But: the converse is true: all fractions of the form $1/2^k$ can be represented in base 10
- Fractional conversions from one base to another are stopped
  - If there is a rational solution or
  - When the desired accuracy is attained

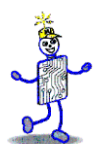
Copyright 2010 John Wiley & Sons, Inc.



Mixed Number Conversion

- Integer and fraction parts must be converted separately
- Radix point: fixed reference for the conversion
  - Digit to the left is a unit digit in every base
  - B^0 is always 1 regardless of the base

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."

CHAPTER 4:
Data Formats

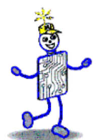
The Architecture of Computer Hardware, Systems Software & Networking:
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University



Data Formats

- Computers
  - Process and store all forms of data in binary format
- Human communication
  - Includes language, images and sounds
- Data formats:
  - Specifications for converting data into computer-usable form
  - Define the different ways human data may be represented, stored and processed by a computer

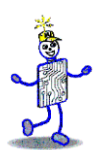
Copyright 2010 John Wiley & Sons, Inc.

Sources of Data

- Binary input
  - Begins as discrete input
  - Example: keyboard input such as ***A 1+2=3 math***
  - Keyboard generates a binary number code for each key
- Analog
  - Continuous data such as sound or images
  - Requires hardware to convert data into binary numbers

Figure 3.1 with this
color scheme

Copyright 2010 John Wiley & Sons, Inc.



Common Data Representations

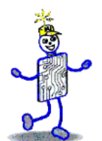
Type of Data	Standard(s)
Alphanumeric	Unicode, ASCII, EDCDIC
Image (bitmapped)	▪GIF (graphical image format) ▪TIF (tagged image file format) ▪PNG (portable network graphics)
Image (object)	PostScript, JPEG, SWF (Macromedia Flash), SVG
Outline graphics and fonts	PostScript, TrueType
Sound	WAV, AVI, MP3, MIDI, WMA
Page description	PDF (Adobe Portable Document Format), HTML, XML
Video	Quicktime, MPEG-2, RealVideo, WMV

Copyright 2010 John Wiley & Sons, Inc.

Internal Data Representation

- Reflects the
  - Complexity of input source
  - Type of processing required
- Trade-offs
  - Accuracy and resolution
    - Simple photo vs. painting in an art book
  - Compactness (storage and transmission)
    - More data required for improved accuracy and resolution
    - *Compression* represents data in a more compact form
    - *Metadata:* data that describes or interprets the meaning of data
  - Ease of manipulation:
    - Processing simple audio vs. high-fidelity sound
  - Standardization
    - *Proprietary formats* for storing and processing data (WordPerfect vs. Word)
    - De facto standards: proprietary standards based on general user acceptance (PostScript)

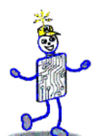
Copyright 2010 John Wiley & Sons, Inc.



Data Types: Numeric

- Used for mathematical manipulation
  - Add, subtract, multiply, divide
- Types
  - Integer (whole number)
  - Real (contains a decimal point)
- Covered in Chapters 4 and 5

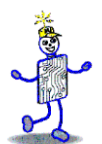
Copyright 2010 John Wiley & Sons, Inc.



Data Types: Alphanumeric

- Alphanumeric:
  - Characters: *b T*
  - Number digits: *7 9*
  - Punctuation marks: *! ;*
  - Special-purpose characters: *$ &*
- Numeric characters vs. numbers
  - Both entered as ordinary characters
  - Computer converts into numbers for calculation
    - Examples: Variables declared as numbers by the programmer (Salary$ in BASIC)
  - Treated as characters if processed as text
    - Examples: Phone numbers, ZIP codes

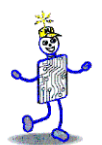
Copyright 2010 John Wiley & Sons, Inc.



Alphanumeric Codes

- Arbitrary choice of bits to represent characters
  - Consistency: input and output device must recognize same code
  - Value of binary number representing character corresponds to placement in the alphabet
    - Facilitates sorting and searching

Copyright 2010 John Wiley & Sons, Inc.



Representing Characters

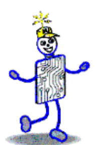
- ASCII - most widely used coding scheme
- EBCDIC: IBM mainframe (legacy)
- Unicode: developed for worldwide use

Copyright 2010 John Wiley & Sons, Inc.

ASCII

- Developed by ANSI (American National Standards Institute)
- Represents
  - Latin alphabet, Arabic numerals, standard punctuation characters
  - Plus small set of accents and other European special characters
- ASCII
  - 7-bit code: 128 characters

Copyright 2010 John Wiley & Sons, Inc.



ASCII Reference Table

LSD \ MSD	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P		p
1	SOH	DC1	!	1	A	Q	a	W
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACJ	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

74_{16}

111 0100

Copyright 2010 John Wiley & Sons, Inc.

EBCDIC

- Extended Binary Coded Decimal Interchange Code developed by IBM
  - Restricted mainly to IBM or IBM compatible mainframes
  - Conversion software to/from ASCII available
  - Common in archival data
  - Character codes differ from ASCII

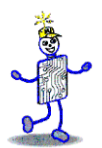
	ASCII	EBCDIC
Space	20_{16}	40_{16}
A	41_{16}	$C1_{16}$
b	62_{16}	82_{16}

Copyright 2010 John Wiley & Sons, Inc.

Unicode

- Most common 16-bit form represents 65,536 characters
- ASCII Latin-I subset of Unicode
  - Values 0 to 255 in Unicode table
- Multilingual: defines codes for
  - Nearly every character-based alphabet
  - Large set of ideographs for Chinese, Japanese and Korean
  - Composite characters for vowels and syllabic clusters required by some languages
- Allows software modifications for local-languages

Copyright 2010 John Wiley & Sons, Inc.

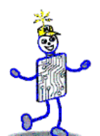


Collating Sequence

- Alphabetic sorting if software handles mixed upper- and lowercase codes
- In ASCII, numbers collate first; in EBCDIC, last
- ASCII collating sequence for string of characters

Letters							
Adam	A	d	a	m			
Adamian	A	d	a	m	i	a	n
Adams	A	d	a	m	s		

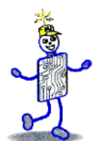
Numeric Characters				
1	011	0001		
12	011	0001	011	0010
2	011	0010		



2 Classes of Codes

- *Printing* characters
  - Produced on the screen or printer
- *Control* characters
  - Control position of output on screen or printer
    - VT: vertical tab
    - LF: Line feed
  - Cause action to occur
    - BEL: bell rings
    - DEL: delete current character
  - Communicate status between computer and I/O device
    - ESC: provides extensions by changing the meaning of a specified number of contiguous following characters

Copyright 2010 John Wiley & Sons, Inc.



Keyboard Input

- *Scan code*
  - Two different scan codes on keyboard
    - One generated when key is struck and another when key is released
  - Converted to Unicode, ASCII or EBCDIC by software in terminal or PC
- Advantage
  - Easily adapted to different languages or keyboard layout
  - Separate scan codes for key press/release for multiple key combinations
    - Examples: shift and control keys

Copyright 2010 John Wiley & Sons, Inc.

Other Alphanumeric Input

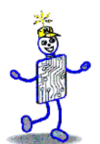
- *OCR* (optical character reader)
  - Scans text and inputs it as character data
  - Used to read specially encoded characters
    - Example: magnetically printed check numbers
- *Bar Code Readers*
  - Used in applications that require fast, accurate and repetitive input with minimal employee training
  - Examples: supermarket checkout counters and inventory control
- *Magnetic stripe reader:* alphanumeric data from credit cards
- *RFID:* store and transmit data between RFID tags and computers
- *Voice*
  - Digitized audio recording common but conversion to alphanumeric data difficult
  - Requires knowledge of sound patterns in a language (*phonemes*) plus rules for pronunciation, grammar, and syntax

Copyright 2010 John Wiley & Sons, Inc.

Image Data

- Photographs, figures, icons, drawings, charts and graphs
- Two approaches:
  - *Bitmap* or *raster images* of photos and paintings with continuous variation
  - *Object* or *vector images* composed of *graphical objects* like lines and curves defined geometrically
- Differences include:
  - Quality of the image
  - Storage space required
  - Time to transmit
  - Ease of modification

Copyright 2010 John Wiley & Sons, Inc.



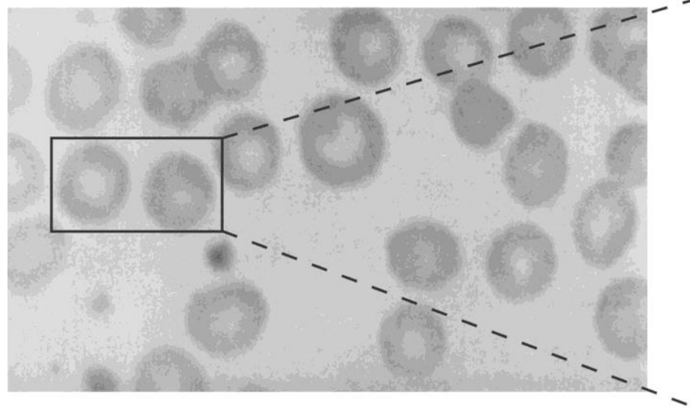
Bitmap Images

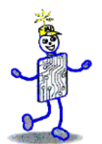
- Used for realistic images with continuous variations in shading, color, shape and texture
  - Examples:
    - Scanned photos
    - Clip art generated by a *paint* program
- Preferred when image contains large amount of detail and processing requirements are fairly simple
- Input devices:
  - Scanners
  - Digital cameras and video capture devices
  - Graphical input devices like mice and pens
- Managed by *photo editing software* or *paint software*
  - Editing tools to make tedious bit by bit process easier

Copyright 2010 John Wiley & Sons, Inc.

Bitmap Images

- Each individual *pixel* (*pi(x)*cture *el*ement) in a graphic stored as a binary number
  - Pixel: A small area with associated coordinate location
  - Example: each point below represented by a 4-bit code corresponding to 1 of 16 shades of gray

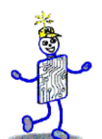




Bitmap Display

- Monochrome: black or white
  - 1 bit per pixel
- Gray scale: black, white or 254 shades of gray
  - 1 byte per pixel
- Color graphics: 16 colors, 256 colors, or 24-bit true color (16.7 million colors)
  - 4, 8, and 24 bits respectively

Copyright 2010 John Wiley & Sons, Inc.



Storing Bitmap Images

- Frequently large files
  - Example: 600 rows of 800 pixels with 1 byte for each of 3 colors ➡ ~1.5MB file
- File size affected by
  - *Resolution* (the number of pixels per inch)
    - Amount of detail affecting clarity and sharpness of an image
  - Levels: number of bits for displaying shades of gray or multiple colors
    - *Palette*: color translation table that uses a code for each pixel rather than actual color value
  - Data compression

Copyright 2010 John Wiley & Sons, Inc.

GIF (Graphics Interchange Format)

- First developed by CompuServe in 1987
- GIF89a enabled animated images
  - allows images to be displayed sequentially at fixed time sequences
- Color limitation: 256
- Image compressed by LZW (Lempel-Zif-Welch) algorithm
- Preferred for line drawings, clip art and pictures with large blocks of solid color
- *Lossless compression*

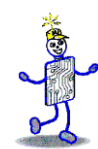
Copyright 2010 John Wiley & Sons, Inc.

GIF (Graphics Interchange Format)

Screen width
Screen
Background
Image 1
Screen height
Image top position
Image 2
Image height
Image left position
Image 3
Image width

Header block "GIFxxa"	Logical screen descriptor block	Global color table	Image 1	Image 2	. . . .
6 bytes	6 bytes	(optional) Up to 768 bytes			

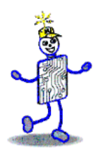
Image descriptor block	Local color table	Image data	
9 bytes	(optional) Up to 768 bytes	Determined from descriptor	(Depends on size of image)



JPEG
(Joint Photographers Expert Group)

- Allows more than 16 million colors
- Suitable for highly detailed photographs and paintings
- Employs *lossy compression* algorithm that
  - Discards data to decreases file size and transmission speed
  - May reduce image resolution, tends to distort sharp lines

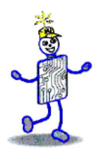
Copyright 2010 John Wiley & Sons, Inc.



Object Images

- Created by *drawing* packages or output from spreadsheet data graphs
- Composed of lines and shapes in various colors
- Computer translates geometric formulas to create the graphic
- Storage space depends on image complexity
  - number of instructions to create lines, shapes, fill patterns
- Movies *Shrek* and *Toy Story* use object images

Copyright 2010 John Wiley & Sons, Inc.



Object Images

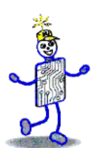
- Based on mathematical formulas
  - Easy to move, scale and rotate without losing shape and identity as bitmap images may
- Require less storage space than bitmap images
- Cannot represent photos or paintings
- Cannot be displayed or printed directly
  - Must be converted to bitmap since output devices except plotters are bitmap

Copyright 2010 John Wiley & Sons, Inc.

PostScript

- *Page description language*: list of procedures and statements that describe each of the objects to be printed on a page
  - Stored in ASCII or Unicode text file
  - Interpreter program in computer or output device reads PostScript to generate image
- Scalable font support
  - Font outline objects specified like other objects

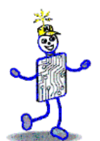
Copyright 2010 John Wiley & Sons, Inc.



Bitmap vs. Object Images

Bitmap (Raster)	Object (Vector)
Pixel map	Geometrically defined shapes
Photographic quality	Complex drawings
Paint software	Drawing software
Larger storage requirements	Higher computational requirements
Enlarging images produces jagged edges	Objects scale smoothly
Resolution of output limited by resolution of image	Resolution of output limited by output device

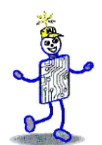
Copyright 2010 John Wiley & Sons, Inc.



Video Images

- Require massive amount of data
  - Video camera producing full screen 640 x 480 pixel true color image at 30 frames/sec → 27.65 MB of data/sec
  - 1-minute film clip → 1.6 GB storage
- Options for reducing file size: decrease size of image, limit number of colors, reduce frame rate
- Method depends on how video delivered to users
  - *Streaming video*: video displayed as it is downloaded from the Web server
  - Local data (file on DVD or downloaded onto system) for higher quality
    - MPEG-2: movie quality images with high compression require substantial processing capability

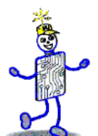
Copyright 2010 John Wiley & Sons, Inc.



Audio Data

- Transmission and processing requirements less demanding than those for video
- *Waveform audio*: digital representation of sound
- *MIDI* (Musical Instrument Digital Interface): instructions to recreate or synthesize sounds
- Analog sound converted to digital values by *A-to-D converter*

Copyright 2010 John Wiley & Sons, Inc.



Waveform Audio

Typical audio waveform

Amplitude

64

48

32

16

0

–16

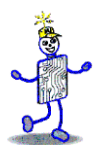
Section expanded

Samples

t

Sampling rate normally 50KHz

Copyright 2010 John Wiley & Sons, Inc.



Sampling Rate

- Number of times per second that sound is measured during the recording process.
  - 1000 samples per second = 1 KHz (kilohertz)
  - Example: Audio CD sampling rate = 44.1KHz
- Height of each sample saved as:
  - 8-bit number for radio-quality recordings
  - 16-bit number for high-fidelity recordings
  - 2 x 16-bits for stereo

Copyright 2010 John Wiley & Sons, Inc.

Audio Formats

- *MP3*
  - Derivative of MPEG-2 (ISO *M*oving *P*icture *E*xperts *Gr*oup)
  - Uses psychoacoustic compression techniques to reduce storage requirements
- *WAV*
  - Developed by Microsoft as part of its multimedia specification
  - General-purpose format for storing and reproducing small snippets of sound

Copyright 2010 John Wiley & Sons, Inc.

Audio Data Formats

MP3 frame MP3 frame MP3 frame

ID Field (optional)	MP3 header	error check (optional)	MP3 data			...		
	32 bytes	2 bytes	384, 576, or 1152 samples					

MP3 file

Chunk id	Length of chunk	Data (<length of chunk> bytes)
4 bytes	4 bytes	

WAVE	Format chunk	Actual sound data
4 bytes	(as needed)	

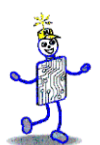
WAV file

Copyright 2010 John Wiley & Sons, Inc.

Data Compression

- *Compression*: recoding data so that it requires fewer bytes of storage space.
- *Compression ratio*: the amount file is shrunk
- *Lossless*: inverse algorithm restores data to exact original form
  - Examples: GIF, PCX, TIFF
- *Lossy*: trades off data degradation for file size and download speed
  - Much higher compression ratios, often 10 to 1
  - Example: JPEG
  - Common in multimedia
- MPEG-2: uses both forms for ratios of 100:1

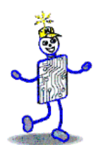
Copyright 2010 John Wiley & Sons, Inc.



Page Description Languages

- Describe layout of objects on a displayed or printed page
- Objects may include text, object images, bitmap images, multimedia objects, and other data formats
- Examples
  - HTML, XHTML, XML
  - PDF
  - Postscript

Copyright 2010 John Wiley & Sons, Inc.



Internal Computer Data Format

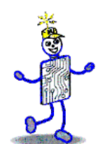
- All data stored as binary numbers
- Interpreted based on
  - Operations computer can perform
  - Data types supported by programming language used to create application

Copyright 2010 John Wiley & Sons, Inc.

5 Simple Data Types

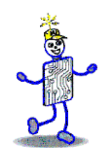
- Boolean: 2-valued variables or constants with values of true or false
- Char: Variable or constant that holds alphanumeric character
- Enumerated
  - User-defined data types with possible values listed in definition
    - Type DayOfWeek = Mon, Tues, Wed, Thurs, Fri, Sat, Sun
- Integer: positive or negative whole numbers
- Real
  - Numbers with a decimal point
  - Numbers whose magnitude, large or small, exceeds computer's capability to store as an integer

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 5: Representing Numerical Data

The Architecture of Computer Hardware and Systems Software & Networking:

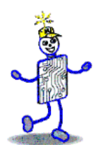
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

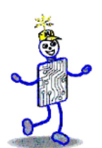
PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University



Number Representation

- Numbers can be represented as a combination of
  - Value or magnitude
  - Sign (plus or minus)
  - Decimal (if necessary)

Copyright 2010 John Wiley & Sons, Inc

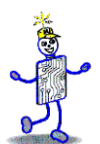


Unsigned Numbers: Integers

- Unsigned whole number or *integer*
- Direct *binary* equivalent of decimal integer
  - 4 bits: 0 to 9
  - 16 bits: 0 to 9,999
  - 8 bits: 0 to 99
  - 32 bits: 0 to 99,999,999

Decimal	Binary	BCD		
68	= 0100 0100 $= 2^6 + 2^2 = 64 + 4 =$ ***68***	= 0110 $= 2^2 + 2^1 =$ ***6***		1000 $2^3 =$ ***8***
99 (largest 8-bit BCD)	= 0110 0011 $= 2^6 + 2^5 + 2^1 + 2^0 =$ $= 64 + 32 + 2 + 1 =$ ***99***	= 1001 $= 2^3 + 2^0$ = ***9***		1001 $2^3 + 2^0$ ***9***
255 (largest 8-bit binary)	= 1111 1111 $= 2^8 - 1 =$ ***255***	= 0010 $= 2^1$ = ***2***	0101 $2^2 + 2^0$ ***5***	0101 $2^2 + 2^0$ ***5***

Copyright 2010 John Wiley & Sons, Inc

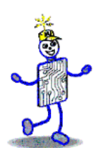


Value Range: Binary vs. BCD

- BCD range of values < conventional binary representation
- Binary: 4 bits can hold 16 different values (0 to 15)
  - BCD: 4 bits can hold only 10 different values (0 to 9)

No. of Bits	BCD Range		Binary Range	
4	0-9	1 digit	0-15	1+ digit
8	0-99	2 digits	0-255	2+ digits
12	0-999	3 digits	0-4,095	3+ digits
16	0-9,999	4 digits	0-65,535	4+ digits
20	0-99,999	5 digits	0-1 million	6 digits
24	0-999,999	6 digits	0-16 million	7+ digits
32	0-99,999,999	8 digits	0-4 billion	9+ digits
64	0-(10^{16}-1)	16 digits	0-16 quintillion	19+ digits

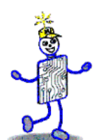
Copyright 2010 John Wiley & Sons, Inc



Conventional Binary vs. BCD

- Binary representation generally preferred
  - Greater range of value for given number of bits
  - Calculations easier
- BCD often used in business applications to maintain decimal rounding and decimal precision

Copyright 2010 John Wiley & Sons, Inc

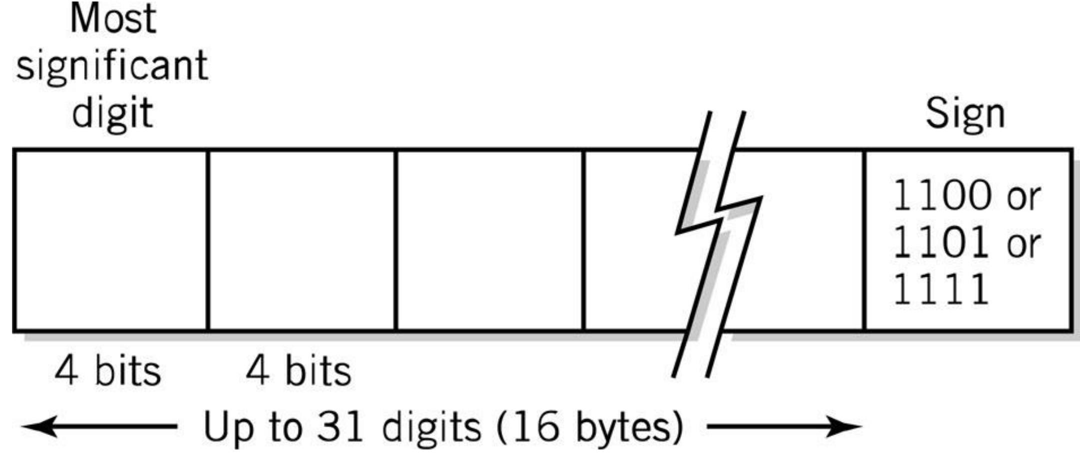


Simple BCD Multiplication

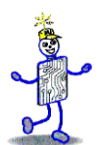
Copyright 2010 John Wiley & Sons, Inc

Packed Decimal Format

- Real numbers representing dollars and cents
- Support by business-oriented languages like COBOL
- IBM System 370/390 and Compaq Alpha



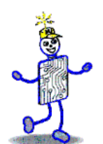
Copyright 2010 John Wiley & Sons, Inc



Signed-Integer Representation

- No obvious direct way to represent the sign in binary notation
- Options:
  - Sign-and-magnitude representation
  - 1’s complement
  - 2’s complement (most common)

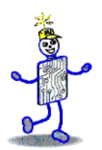
Copyright 2010 John Wiley & Sons, Inc



Sign-and-Magnitude

- Use left-most bit for sign
  - 0 = plus; 1 = minus
- Total range of integers the same
  - Half of integers positive; half negative
  - Magnitude of largest integer half as large
- Example using 8 bits:
  - Unsigned: 1111 1111 = +255
  - Signed: 0111 1111 = +127
1111 1111 = -127
  - Note: 2 values for 0:
+0 (0000 0000) and -0 (1000 0000)

Copyright 2010 John Wiley & Sons, Inc

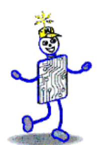


Difficult Calculation Algorithms

- Sign-and-magnitude algorithms complex and difficult to implement in hardware
  - Must test for 2 values of 0
  - Useful with BCD
  - Order of signed number and carry/borrow makes a difference
- Example: Decimal addition algorithm

Addition: 2 Positive Numbers	Addition: 1 Signed Number		
4 +2 6	4 - 2 2	2 - 4 -2	12 - 4 8

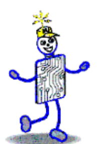
Copyright 2010 John Wiley & Sons, Inc



Complementary Representation

- Sign of the number does not have to be handled separately
- Consistent for all different signed combinations of input numbers
- Two methods
  - Radix: value used is the base number
  - Diminished radix: value used is the base number minus 1
    - 9's complement: base 10 diminished radix
    - 1's complement: base 2 diminished radix

Copyright 2010 John Wiley & Sons, Inc



9's Decimal Complement

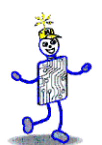
- *Taking the complement*: subtracting a value from a standard basis value
- Decimal (base 10) system diminished radix complement
  - Radix minus 1 = 10 – 1 → 9 as the basis
  - 3-digit example: base value = 999
  - Range of possible values 0 to 999 arbitrarily split at 500

Numbers	Negative		Positive	
Representation method	Complement		Number itself	
Range of decimal numbers	***-499***	***-000***	***+0***	***499***
Calculation	999 minus number		none	
Representation example	***500***	***999***	***0***	***499***

999 – 499 (→ 500)

– Increasing value + →

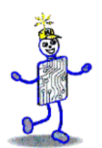
Copyright 2010 John Wiley & Sons, Inc



9's Decimal Complement

- Necessary to specify number of digits or *word size*
- Example: representation of 3-digit number
  - First digit = 0 through 4 ➡ positive number
  - First digit = 5 through 9 ➡ negative number
- Conversion to sign-and-magnitude number for 9's complement
  - **321** remains **321**
  - **521**: take the complement (999 – 521) = **– 478**

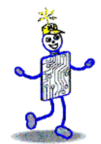
Copyright 2010 John Wiley & Sons, Inc



Choice of Representation

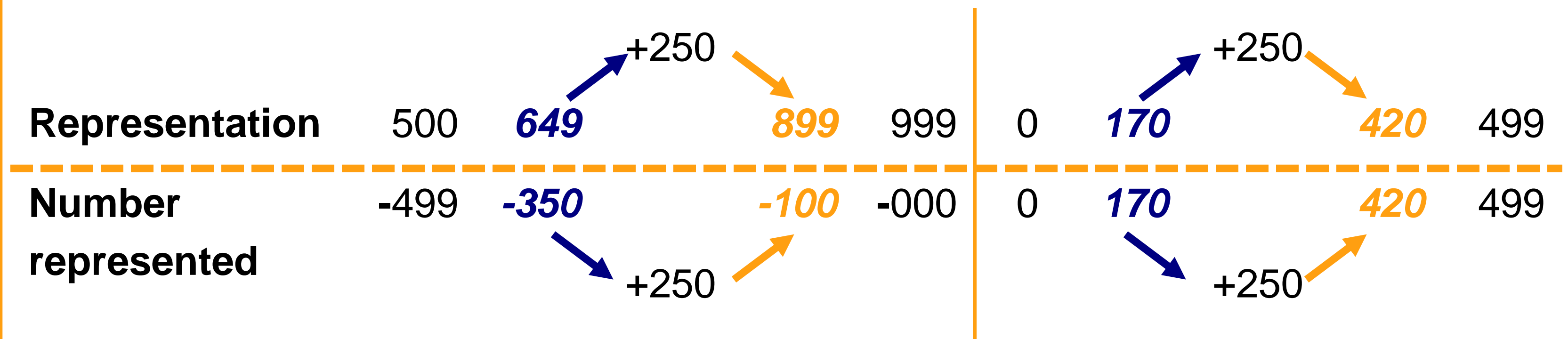
- Must be consistent with rules of normal arithmetic
  - - (-value) = value
- If we complement the value twice, it should return to its original value
  - Complement = basis – value
  - Complement twice
    - Basis – (basis – value) = value

Copyright 2010 John Wiley & Sons, Inc



Modular Addition

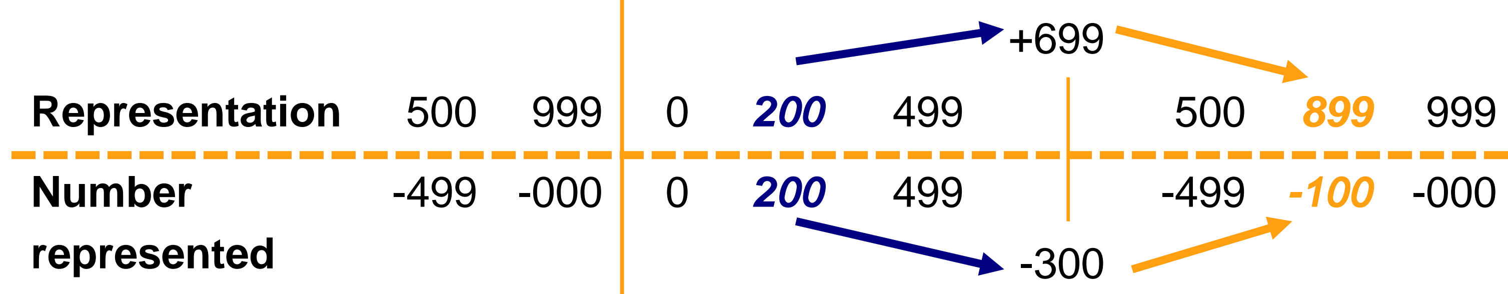
- Counting upward on scale corresponds to addition
- Example in 9's complement: does not cross the modulus

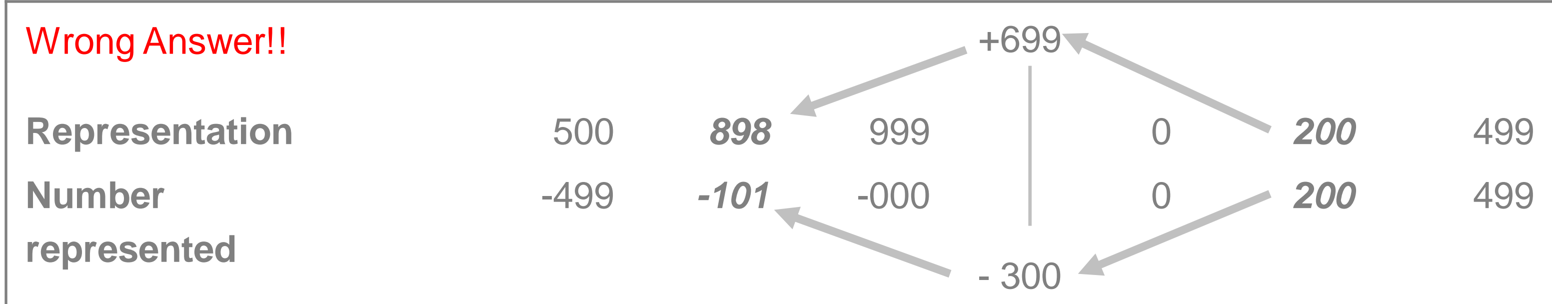


Copyright 2010 John Wiley & Sons, Inc

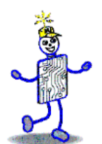
Addition with Wraparound

- Count to the right to add a negative number
- *Wraparound* scale used to extend the range for the negative result
  - Counting left would cross the modulus and give incorrect answer because there are 2 values for 0 (+0 and -0)



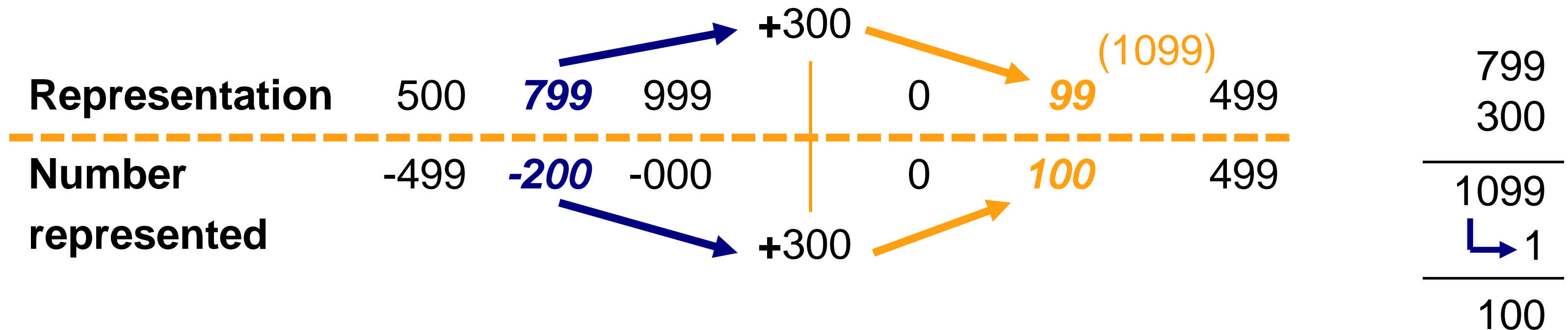


Copyright 2010 John Wiley & Sons, Inc

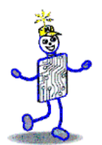


Addition with End-around Carry

- Count to the right crosses the modulus
- End-around carry
  - Add 2 numbers in 9's complementary arithmetic
  - If the result has more digits than specified, add carry to the result



Copyright 2010 John Wiley & Sons, Inc



Overflow

- Fixed word size has a fixed range size
- Overflow: combination of numbers that adds to result outside the range
- End-around carry in modular arithmetic avoids problem
- Complementary arithmetic: numbers *out of range* have the opposite sign
  - Test: If both inputs to an addition have the same sign and the output sign is different, an overflow occurred

Copyright 2010 John Wiley & Sons, Inc

1's Binary Complement

- *Taking the complement*: subtracting a value from a standard basis value
  - Binary (base 2) system diminished radix complement
  - Radix minus 1 = 2 – 1 → 1 as the basis
- *Inversion: change 1's to 0's and 0's to 1s*
  - Numbers beginning with 0 are positive
  - Numbers beginning with 1 are negative
  - 2 values for zero
- Example with 8-bit binary numbers

Numbers	Negative		Positive	
Representation method	Complement		Number itself	
Range of decimal numbers	-127_{10}	-0_{10}	$+0_{10}$	127_{10}
Calculation	Inversion		None	
Representation example	10000000	11111111	00000000	01111111

Copyright 2010 John Wiley & Sons, Inc

Conversion between Complementary Forms

- Cannot convert directly between 9's complement and 1's complement
  - Modulus in 3-digit decimal: 999
    - Positive range 499
  - Modulus in 8-bit binary: 11111111 or 255_{10}
    - Positive range 01111111 or 127_{10}
- Intermediate step: sign-and-magnitude representation

Copyright 2010 John Wiley & Sons, Inc

Addition

- Add 2 positive 8-bit numbers
- Add 2 8-bit numbers with different signs
  - Take the 1's complement of 58 (i.e., invert)
    0011 1010
    1100 0101

0010 1101	=	45
0011 1010	=	58
0110 0111	=	103

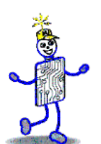
0010 1101	=	45
1100 0101	=	–58
1111 0010	=	**–13**

Invert to get magnitude

0000 1101

8 + 4 + 1 = 13

Copyright 2010 John Wiley & Sons, Inc

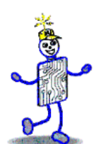


Addition with Carry

- 8-bit number
  - Invert
    0000 0010 (2_{10})
    1111 1101
  - Add
  - 9 bits

End-around carry

0110 1010	=	106
1111 1101	=	–2
10110 0111		
+1		
0110 1000	=	104



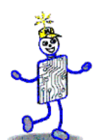
Subtraction

- 8-bit number
  - Invert
    0101 1010 (90_{10})
    1010 0101
  - Add
  - 9 bits

    End-around carry

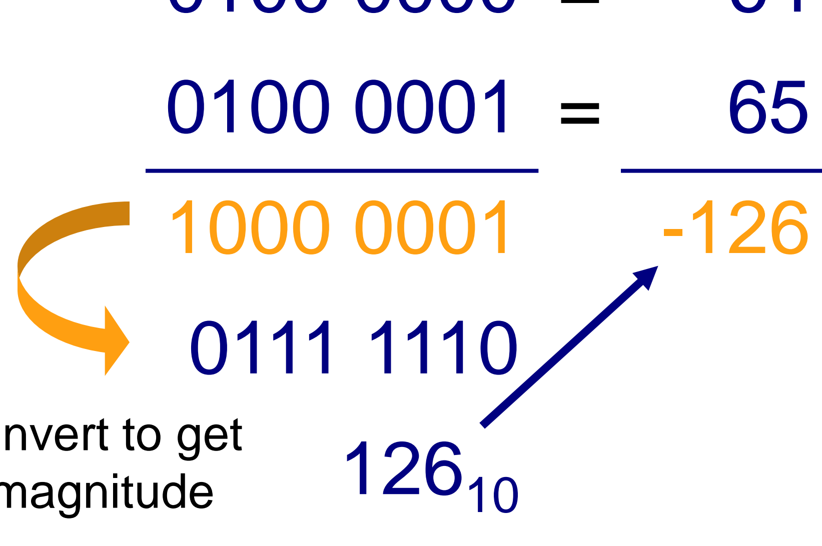
0110 1010	=	106
-0101 1010	=	90
0110 1010	=	106
–1010 0101	=	90
10000 1111		
+1		
0001 0000	=	16

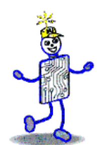
Copyright 2010 John Wiley & Sons, Inc



Overflow

- 8-bit number
  - 256 different numbers
  - Positive numbers: 0 to 127
- Add
  - Test for *overflow*
  - 2 positive inputs produced negative result → *overflow!*
  - Wrong answer!
- Programmers beware: some high-level languages, e.g., some versions of BASIC, do not check for overflow adequately



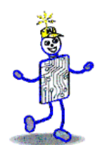


10's Complement

- Create complementary system with a single 0
- Radix complement: use the base for complementary operations
  - Decimal base: 10's complement
  - Example: Modulus 1000 as the as reflection point

Numbers	Negative		Positive	
Representation method	Complement		Number itself	
Range of decimal numbers	***-500***	***-001***	***0***	***499***
Calculation	1000 minus number		none	
Representation example	***500***	***999***	***0***	***499***

Copyright 2010 John Wiley & Sons, Inc



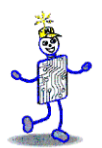
Examples with 3-Digit Numbers

- Example 1:
  - 10's *complement representation* of 247
    - 247 (positive number)
  - 10's *complement* of 227
    - 1000 – 247 = 753 (negative number)
- Example 2:
  - 10's complement of 17
    - 1000 – 017 = 983
- Example 3:
  - 10's complement of 777
    - Negative number because first digit is 7
    - 1000 – 777 = 223
    - Signed value = -223

Copyright 2010 John Wiley & Sons, Inc

Alternative Method for 10's Complement

- Based on 9's complement
- Example using 3-digit number
  - Note: 1000 = 999 + 1
  - 9's complement = 999 – value
  - Rewriting
    - 10's complement = 1000 – value = 999 + 1 – value
  - Or: 10's complement = 9's complement + 1
- Computationally easier especially when working with binary numbers

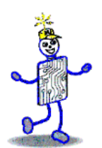


2's Complement

- Modulus = a base 2 "1" followed by specified number of 0's
  - For 8 bits, the modulus = 1000 0000
- Two ways to find the complement
  - Subtract value from the modulus or invert

Numbers	Negative		Positive	
Representation method	Complement		Number itself	
Range of decimal numbers	-128_{10}	-1_{10}	$+0_{10}$	127_{10}
Calculation	Inversion		None	
Representation example	10000000	11111111	00000000	01111111

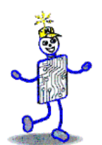
Copyright 2010 John Wiley & Sons, Inc



Estimating Integer Size

- Positive numbers begin with 0
- Small negative numbers (close to 0) begin with multiple 0's
  - 1111 1110 = -2 in 8-bit 2's complements
  - 1000 0000 = -128, largest negative 2's complements
  - Invert all 1's and 0's and approximate the value

Copyright 2010 John Wiley & Sons, Inc



Overflow and Carry Conditions

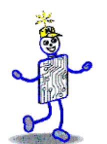
- *Carry flag*: set when the result of an addition or subtraction exceeds fixed number of bits allocated
- *Overflow*: result of addition or subtraction overflows into the sign bit

Copyright 2010 John Wiley & Sons, Inc

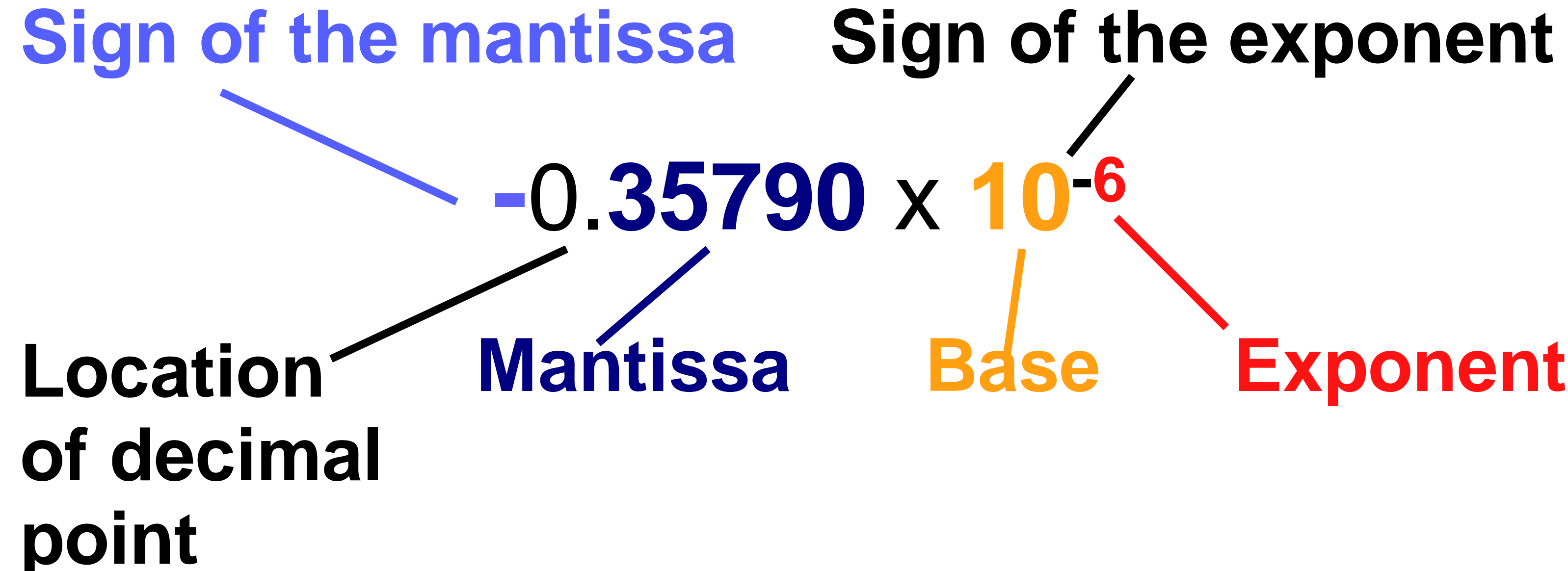
Exponential Notation

- Also called *scientific notation*
  - 12345
  - 12345×10^0
  - 0.12345×10^5
  - 123450000×10^{-4}
- 4 specifications required for a number
  1. Sign ("+" in example)
  2. Magnitude or *mantissa* (12345)
  3. Sign of the exponent ("+" in 10^5)
  4. Magnitude of the exponent (5)
- Plus
  5. Base of the exponent (10)
  6. Location of decimal point (or other base) radix point

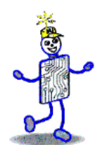
Copyright 2010 John Wiley & Sons, Inc



Summary of Rules



Copyright 2010 John Wiley & Sons, Inc



Format Specification

- Predefined format, usually in 8 bits
  - Increased range of values (two digits of exponent) traded for decreased precision (two digits of mantissa)

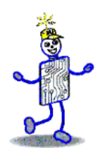
Sign of the mantissa

SEEMMMMM

2-digit Exponent

5-digit Mantissa

Copyright 2010 John Wiley & Sons, Inc

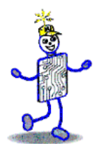


Format

- Mantissa: sign digit in sign-magnitude format
- Assume decimal point located at beginning of mantissa
- Excess-N notation: Complementary notation
  - Pick middle value as offset where N is the middle value

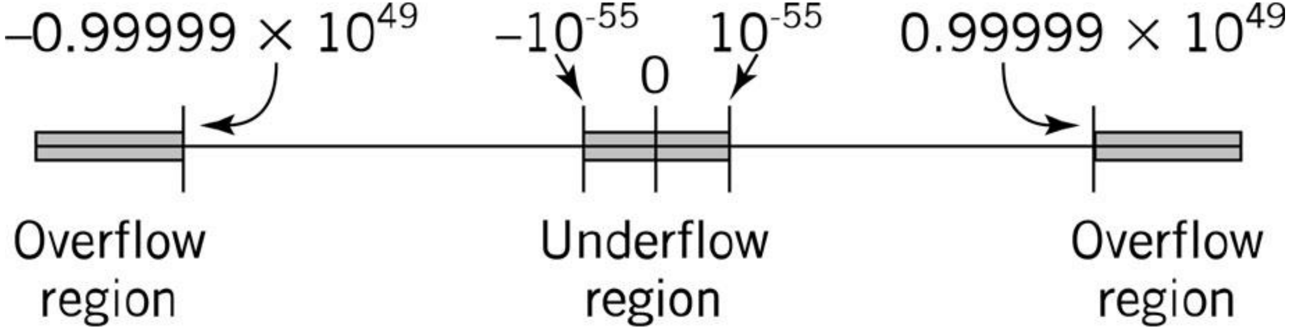
Representation	0	49	50	99
Exponent being represented	-50	-1	0	49

– Increasing value +

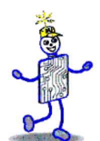


Overflow and Underflow

- Possible for the number to be too large or too small for representation



Copyright 2010 John Wiley & Sons, Inc



Floating Point Calculations

- Addition and subtraction
  - Exponent and mantissa treated separately
  - Exponents of numbers must agree
    - Align decimal points
    - Least significant digits may be lost
  - Mantissa overflow requires exponent again shifted right

Copyright 2010 John Wiley & Sons, Inc

Addition and Subtraction

Add 2 floating point numbers	05199520
	+ 04967850
Align exponents	05199520
	0510067850
Add mantissas; (1) indicates a carry	(1)0019850
Carry requires right shift	05210019(850)
Round	05210020
Check results	

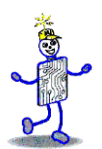
$$05199520 = 0.99520 \times 10^{1} = 9.9520$$

$$04967850 = 0.67850 \times 10^{-1} = 0.06785$$

$$= 10.01985$$

In exponential form $= 0.1001985 \times 10^{2}$

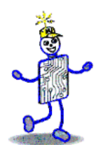
Copyright 2010 John Wiley & Sons, Inc



Multiplication and Division

- Mantissas: multiplied or divided
- Exponents: added or subtracted
  - Normalization necessary to
    - Restore location of decimal point
    - Maintain precision of the result
  - Adjust excess value since added twice
    - Example: 2 numbers with exponent = 3 represented in excess-50 notation
    - 53 + 53 =106
    - Since 50 added twice, subtract: 106 – 50 =56

Copyright 2010 John Wiley & Sons, Inc



Multiplication and Division

- Maintaining precision:
  - Normalizing and rounding multiplication
    - Multiply 2 numbers — 05220000 x 04712500
    - Add exponents, subtract offset — 52 + 47 – 50 = 49
    - Multiply mantissas — 0.20000 x 0.12500 = 0.025000000
    - Normalize the results — 04825000
    - Round — 05210020
    - Check results

05220000 = 0.20000 x 10^{2}

04712500 = 0.125 x 10^{-3}

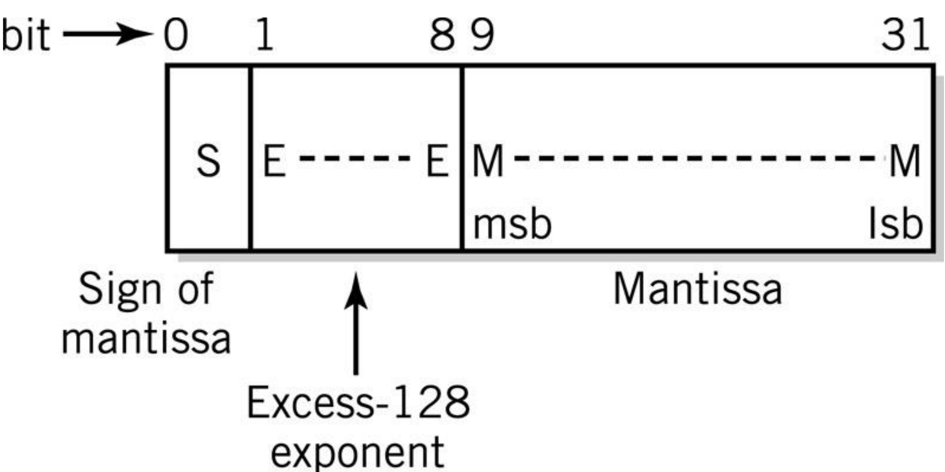
= 0.0250000000 x 10^{-1}

- Normalizing and rounding = 0.25000 x 10^{-2}

Copyright 2010 John Wiley & Sons, Inc

Floating Point in the Computer

- Typical floating point format
  - 32 bits provide range ~10^{-38} to 10^{+38}
  - 8-bit exponent = 256 levels
    - Excess-128 notation
  - 23/24 bits of mantissa: approximately 7 decimal digits of precision



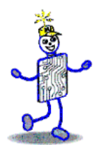
Copyright 2010 John Wiley & Sons, Inc

IEEE 754 Standard

- 32-bit Floating Point Value Definition

Exponent	Mantissa	Value
0	±0	0
0	Not 0	$\pm 2^{-126} \times 0.M$
1^{-254}	Any	$\pm 2^{-127} \times 1.M$
255	±0	±∞
255	not 0	special condition

Copyright 2010 John Wiley & Sons, Inc



Conversion: Base 10 and Base 2

- Two steps
  - Whole and fractional parts of numbers with an embedded decimal or binary point must be converted separately
  - Numbers in exponential form must be reduced to a pure decimal or binary mixed number or fraction before the conversion can be performed

Copyright 2010 John Wiley & Sons, Inc

Conversion: Base 10 and Base 2

- Convert 253.75_{10} to binary floating point form

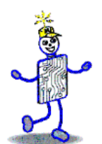
- Multiply number by 100 25375
- Convert to binary equivalent 110 0011 0001 1111 or 1.1000 1100 0111 11 x 2^{14}
- IEEE Representation 0 10001101 10001100011111

Sign

Excess-127
Exponent = 127 + 14

Mantissa

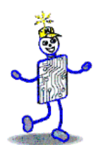
- Divide by binary floating point equivalent of 100_{10} to restore original decimal value



Programming Considerations

- Integer advantages
  - Easier for computer to perform
  - Potential for higher precision
  - Faster to execute
  - Fewer storage locations to save time and space
- Most high-level languages provide 2 or more formats
  - Short integer (16 bits)
  - Long integer (64 bits)

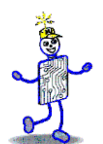
Copyright 2010 John Wiley & Sons, Inc



Programming Considerations

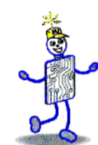
- Real numbers
  - Variable or constant has fractional part
  - Numbers take on very large or very small values outside integer range
  - Program should use least precision sufficient for the task
  - Packed decimal attractive alternative for business applications

Copyright 2010 John Wiley & Sons, Inc



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 6: The Little Man Computer

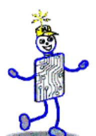
The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley University



The Little Man Computer

Calculator
123
Mailboxes
00 500
01 199
02 500
03 399
95
96
97
98
99 123
In Basket
Out Basket
05
Reset button
Instruction location counter
Little man

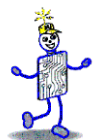
Copyright 2010 John Wiley & Sons, Inc.

Mailboxes: Address vs. Content

- Addresses are consecutive starting at 00 and ending at 99
- Content may be
  - Data, a three digit number, or
  - Instructions

Address	Content

Copyright 2010 John Wiley & Sons, Inc.



Content: Instructions

- Op code
  - In LMC, represented by a single digit
  - Operation code
  - Arbitrary mnemonic
- Operand
  - In LMC, represented by two digits following the op code
  - Object to be manipulated
    - Data or
    - Address of data

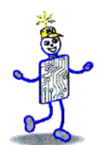
Address	Content	
	Op code	Operand

Copyright 2010 John Wiley & Sons, Inc.

Magic!

- Load program into memory
- Put data into In Basket

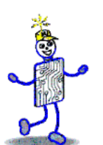
Copyright 2010 John Wiley & Sons, Inc.



Assembly Language

- Specific to a CPU
- 1 to 1 correspondence between assembly language instruction and binary (machine) language instruction
- *Mnemonics* (short character sequence) represent instructions
- Used when programmer needs precise control over hardware, e.g., device drivers

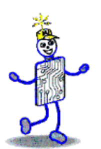
Copyright 2010 John Wiley & Sons, Inc.



Instruction Set

Arithmetic	1xx	ADD
	2xx	SUB
Data Movement	3xx	STORE
	5xx	LOAD
Input/Output	901	INPUT
	902	Output
Machine Control (coffee break)	000	HALT COB

Copyright 2010 John Wiley & Sons, Inc.

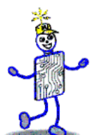


Input/Output

- Move data between calculator and in/out baskets

	Content	
	Op Code	Operand (address)
IN (input)	9	01
OUT (output)	9	02

Copyright 2010 John Wiley & Sons, Inc.



LMC Input/Output

Calculator

Mailboxes

00	500
01	199
02	500
03	399
95	
96	
97	
98	
99	123

123

In Basket

IN

OUT

Out Basket

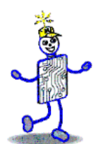
05

Reset button

Instruction location counter

Little man

Copyright 2010 John Wiley & Sons, Inc.



Internal Data Movement

- Between mailbox and calculator

	Content	
	Op Code	Operand (address)
STO (store)	3	xx
LDA (load)	5	xx

Copyright 2010 John Wiley & Sons, Inc.

LMC Internal Data

Calculator

123

Mailboxes

00	500
01	199
02	500
03	399
95	
96	
97	
98	
99	123

LDA

STO

In Basket

Out Basket

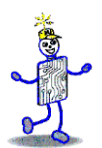
05

Reset button

Instruction location counter

Little man

Copyright 2010 John Wiley & Sons, Inc.

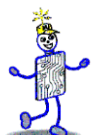


Arithmetic Instructions

- Read mailbox
- Perform operation in the calculator

	Content	
	Op Code	Operand (address)
ADD	1	xx
SUB	2	xx

Copyright 2010 John Wiley & Sons, Inc.



LMC Arithmetic Instructions

Calculator

Mailboxes

123

00	500
01	199
02	500
03	399
95	
96	
97	
98	
99	123

ADD

SUB

In Basket

Out Basket

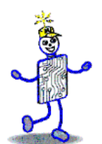
05

Reset button

Instruction location counter

Little man

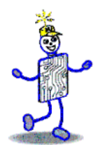
Copyright 2010 John Wiley & Sons, Inc.



Data storage location

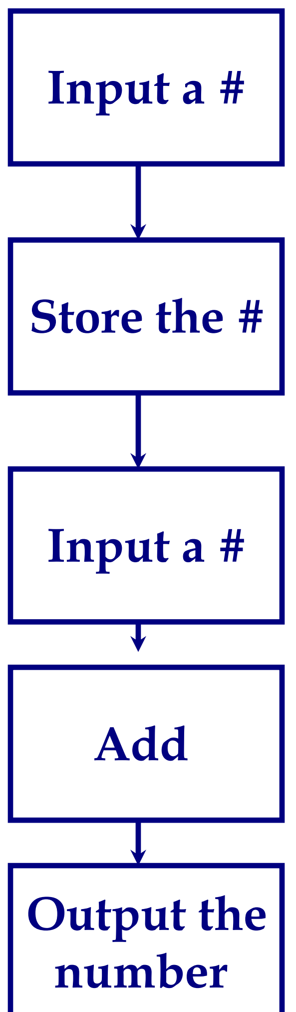
- Physically identical to instruction mailbox
- Not located in instruction sequence
- Identified by *DAT* mnemonic

Copyright 2010 John Wiley & Sons, Inc.

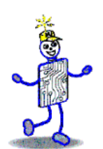


Simple Program: Add 2 Numbers

- Assume data is stored in mailboxes with addresses >90
- Write instructions



Copyright 2010 John Wiley & Sons, Inc.



Program to Add 2 Numbers

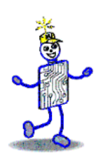
Mailbox	Code	Instruction Description
00	901	;input 1st Number
01	399	;store data
02	901	;input 2nd Number
03	199	;add 1st # to 2nd #
04	902	;output result
05	000	;stop
99	000	;data

Copyright 2010 John Wiley & Sons, Inc.

Program to Add 2 Numbers: Using Mnemonics

Mailbox	Mnemonic	Instruction Description
00	IN	;input 1st Number
01	STO 99	;store data
02	IN	;input 2nd Number
03	ADD 99	;add 1st # to 2nd #
04	OUT	;output result
05	COB	;stop
99	DAT 00	;data

Copyright 2010 John Wiley & Sons, Inc.

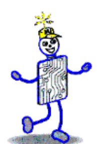


Program Control

- Branching (executing an instruction out of sequence)
  - Changes the address in the counter
- Halt

	Content	
	Op Code	Operand (address)
BR (Jump)	6	xx
BRZ (Branch on 0)	7	xx
BRP (Branch on +)	8	xx
COB (stop)	0	(ignore)

Copyright 2010 John Wiley & Sons, Inc.



LMC Instruction Set

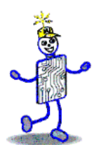
Arithmetic	1xx	ADD
	2xx	SUB
Data Movement	3xx	STORE
	5xx	LOAD
BR	6xx	JUMP
BRZ	7xx	BRANC ON 0
BRP	8xx	BRANCH ON +
Input/Output	901	INPUT
	902	OUTPUT
Machine Control (coffee break)	000	HALT COB

Copyright 2010 John Wiley & Sons, Inc.

Find Positive Difference of 2 Numbers

00	IN	901	
01	STO 10	310	
02	IN	901	
03	STO 11	311	
04	SUB 10	210	
05	BRP 08	808	;test
06	LDA 10	510	;if negative, reverse order
07	SUB 11	211	
08	OUT	902	;print result and
09	COB	000	;stop
10	DAT 00	000	;used for data
11	DAT 00	000	;used for data

Copyright 2010 John Wiley & Sons, Inc.

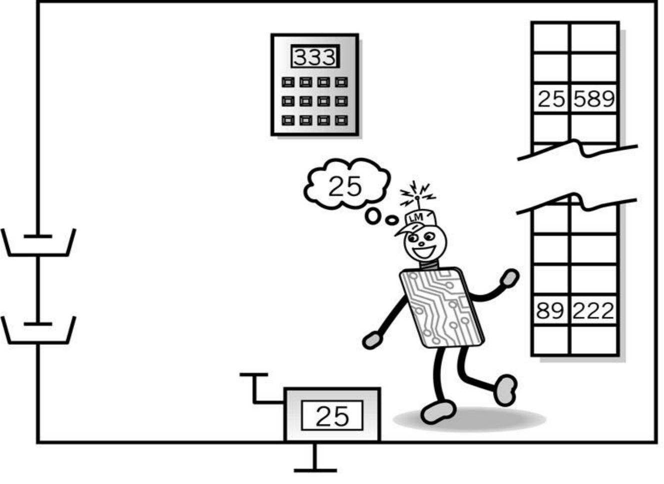


Instruction Cycle

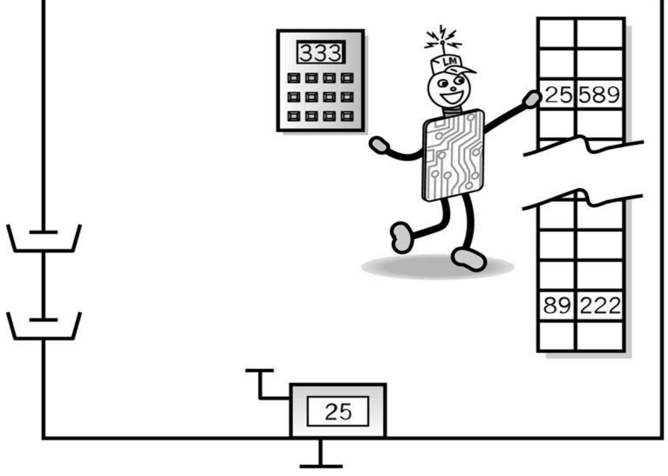
- *Fetch*: Little Man finds out what instruction he is to execute
- *Execute*: Little Man performs the work.

Copyright 2010 John Wiley & Sons, Inc.

Fetch Portion of Fetch and Execute Cycle



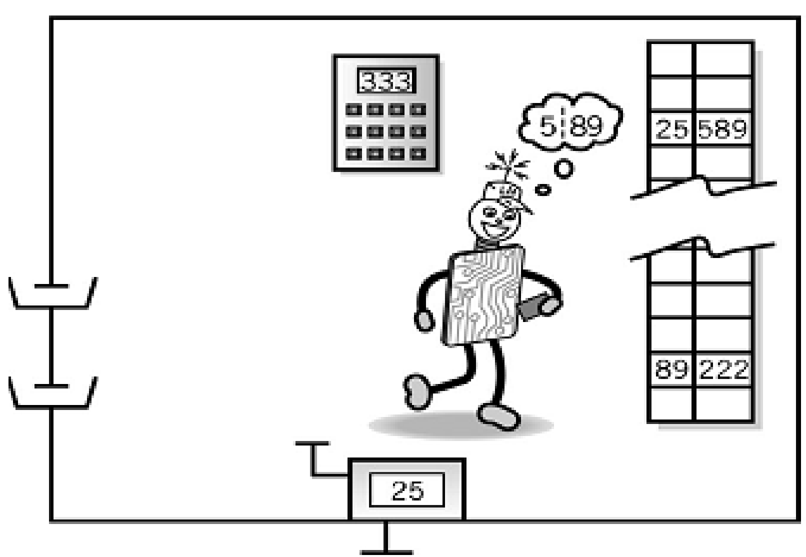
1. Little Man reads the address from the location counter



2. He walks over to the mailbox that corresponds to the location counter

Copyright 2010 John Wiley & Sons, Inc.

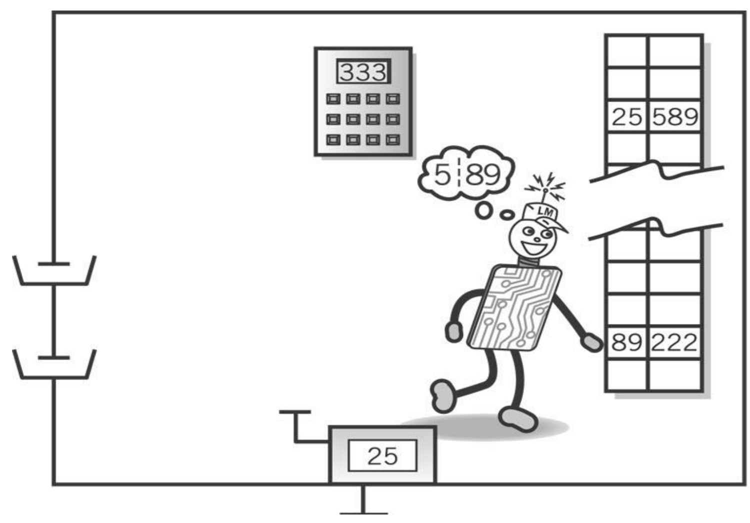
Fetch, cont.



3. And reads the number on the slip of paper (he puts the slip back in case he needs to read it again later)

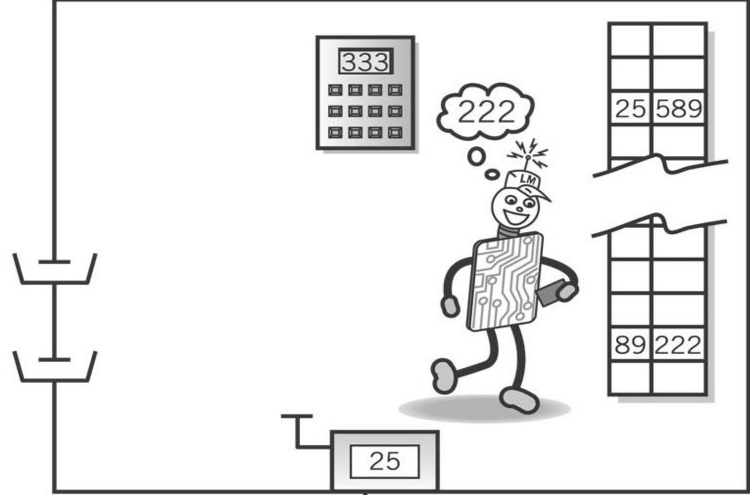
Copyright 2010 John Wiley & Sons, Inc.

Execute Portion

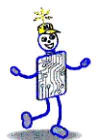


1. The Little Man goes to the mailbox address specified in the instruction he just fetched.

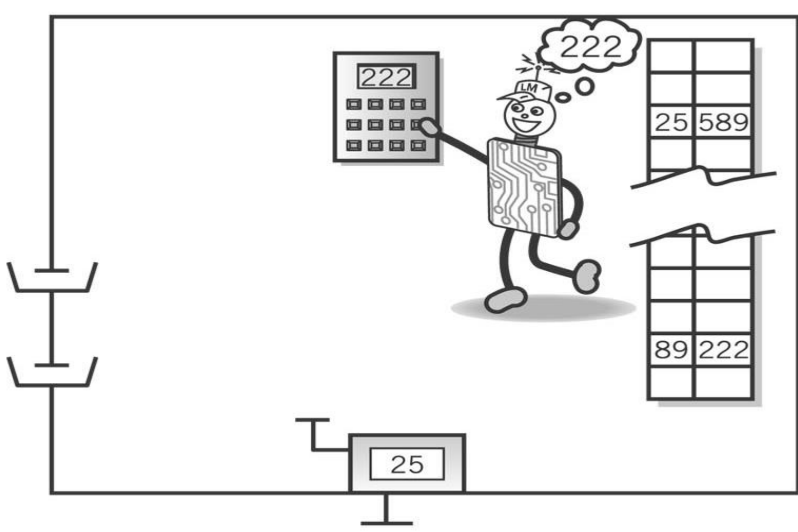
2. He reads the number in that mailbox (he remembers to replace it in case he needs it later).



Copyright 2010 John Wiley & Sons, Inc.



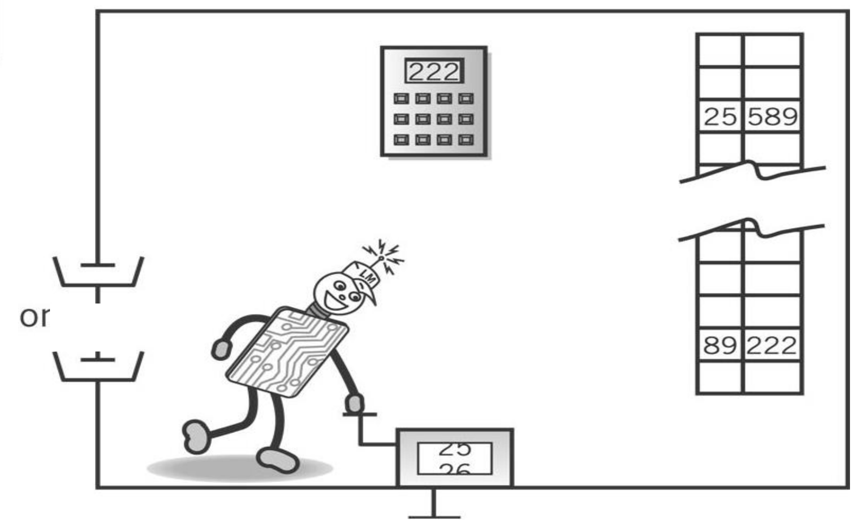
Execute, cont.



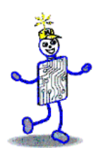
3. He walks over to the calculator and punches the number in.

4. He walks over to the location counter and clicks it, which gets him ready to fetch the next instruction.

or



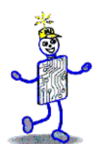
Copyright 2010 John Wiley & Sons, Inc.



von Neumann Architecture (1945)

- Stored program concept
- Memory is addressed linearly
- Memory is addressed without regard to content

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein.”

CHAPTER 7: The CPU and Memory

The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

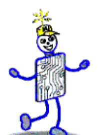
PowerPoint slides authored by Wilson Wong, Bentley University

PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley College

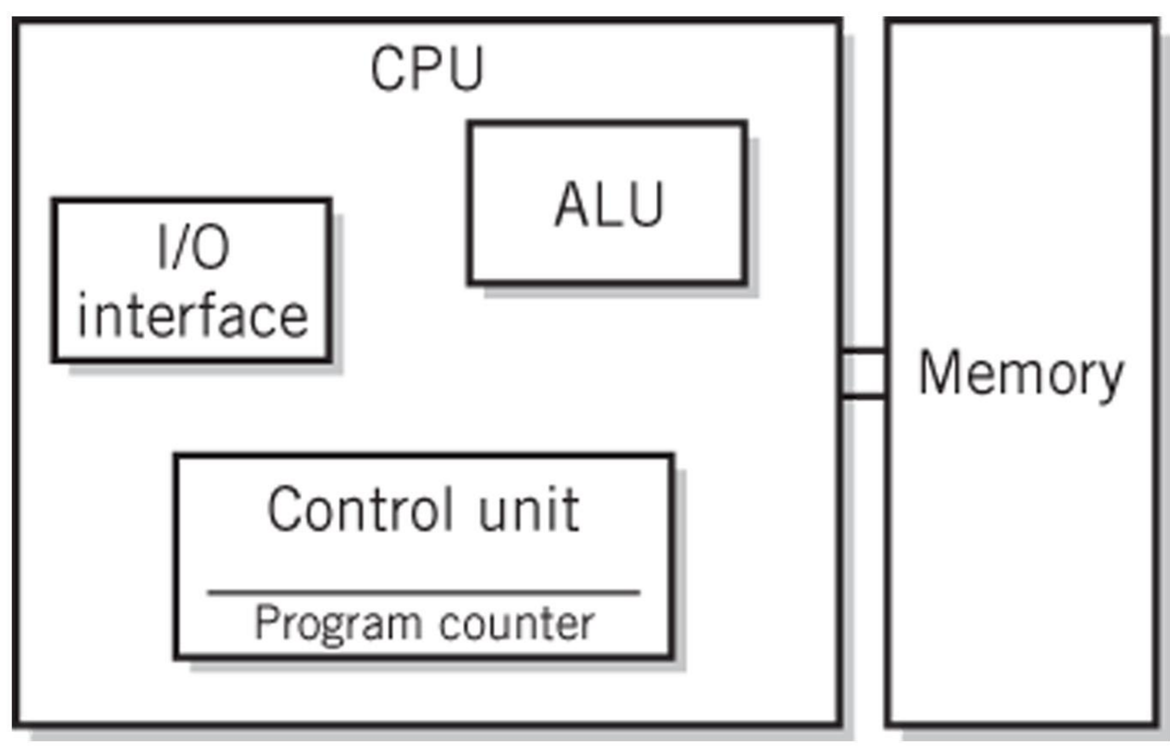
CPU: Major Components

- *ALU* (arithmetic logic unit)
  - Performs calculations and comparisons
- *CU* (control unit)
  - Performs fetch/execute cycle
    - Accesses program instructions and issues commands to the ALU
    - Moves data to and from CPU registers and other hardware components
  - Subcomponents:
    - *Memory management unit:* supervises fetching instructions and data from memory
    - *I/O Interface:* sometimes combined with memory management unit as *Bus Interface Unit*

Copyright 2010 John Wiley & Sons, Inc.



System Block Diagram



Copyright 2010 John Wiley & Sons, Inc.

The Little Man Computer

ALU
123

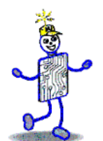
00 500
01 199
02 500
03 399
95
96
97
98
99 123
Memory

I/O
interface

Control
unit

05
Program
counter

Copyright 2010 John Wiley & Sons, Inc.



Concept of Registers

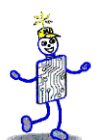
- Small, *permanent* storage locations within the CPU used for a particular purpose
- Manipulated directly by the Control Unit
- Wired for *specific function*
- Size in bits or bytes (not in MB like memory)
- Can hold data, an address or an instruction
- How many registers does the LMC have?
- What are the registers in the LMC?

Copyright 2010 John Wiley & Sons, Inc.

Registers

- Use of Registers
  - Scratchpad for currently executing program
    - Holds data needed quickly or frequently
  - Stores information about status of CPU and currently executing program
    - Address of next program instruction
    - Signals from external devices
- General Purpose Registers
  - *User-visible registers*
  - Hold intermediate results or data values, e.g., loop counters
  - Equivalent to LMC's calculator
  - Typically several dozen in current CPUs

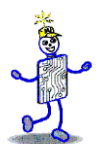
Copyright 2010 John Wiley & Sons, Inc.



Special-Purpose Registers

- *Program Count Register* (PC)
  - Also called instruction pointer
- *Instruction Register (IR)*
  - Stores instruction fetched from memory
- *Memory Address Register (MAR)*
- *Memory Data Register (MDR)*
- *Status Registers*
  - Status of CPU and currently executing program
  - *Flags* (one bit Boolean variable) to track condition like arithmetic carry and overflow, power failure, internal computer error

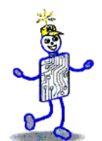
Copyright 2010 John Wiley & Sons, Inc.



Register Operations

- Stores values from other locations (registers and memory)
- Addition and subtraction
- Shift or rotate data
- Test contents for conditions such as zero or positive

Copyright 2010 John Wiley & Sons, Inc.



Operation of Memory

- Each memory location has a unique address
- Address from an instruction is copied to the MAR which finds the location in memory
- CPU determines if it is a store or retrieval
- Transfer takes place between the MDR and memory
- MDR is a two way register

Copyright 2010 John Wiley & Sons, Inc.

Relationship between MAR, MDR and Memory

Address

Data

a byte

memory address register

bit 0

bit n

address decoder

000

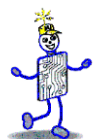
001

$2^n - 1$

address line

individual memory cells

memory data register



MAR-MDR Example

Memory address register

msb

1
1
0
0
0
1

lsb

$110001_2 = 49_{10}$

Address decoder

0
1
49
63

Active line

Memory data register

Copyright 2010 John Wiley & Sons, Inc.

Visual Analogy of Memory

Memory address register

msb

1
1
0
0
0
1

isb

Address decoder

0
1
49
63

All cells dark

1 0 1 0 1 1 0 1

$110001_2 = 49_{10}$

10101101

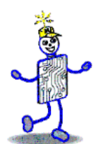
Memory data register

Copyright 2010 John Wiley & Sons, Inc.

Individual Memory Cell

address line = “1”
activate line = “1”
R/W line = “1” (read)
AND
READ SWITCH, R
ONE MEMORY CELL
Data read when READ SWITCH is ON
Data written when WRITE SWITCH is ON
address line = “1”
activate line = “1”
R/W line = “0” (write)
AND
WRITE SWITCH, W
MDR line

Copyright 2010 John Wiley & Sons, Inc.

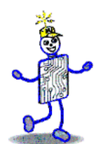


Memory Capacity

Determined by two factors

1. Number of bits in the MAR
   - LMC = 100 (00 to 99)
   - 2^K where K = width of the register in bits
2. Size of the address portion of the instruction
   - 4 bits allows 16 locations
   - 8 bits allows 256 locations
   - 32 bits allows 4,294,967,296 or 4 GB

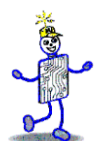
Copyright 2010 John Wiley & Sons, Inc.



RAM: Random Access Memory

- *DRAM (Dynamic RAM)*
  - Most common, cheap, less electrical power, less heat, smaller space
  - Volatile: must be refreshed (recharged with power) 1000's of times each second
- *SRAM (static RAM)*
  - Faster and more expensive than DRAM
  - Volatile
  - Small amounts are often used in *cache memory* for high-speed memory access

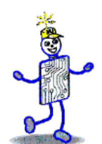
Copyright 2010 John Wiley & Sons, Inc.



Nonvolatile Memory

- *ROM*
  - Read-only Memory
  - Holds software that is not expected to change over the life of the system
- EEPROM
  - Electrically Erasable Programmable ROM
- *Flash Memory*
  - Faster than disks but more expensive
  - Uses hot carrier injection to store bits of data
  - Slow rewrite time compared to RAM
  - Useful for nonvolatile portable computer storage

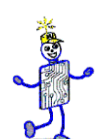
Copyright 2010 John Wiley & Sons, Inc.



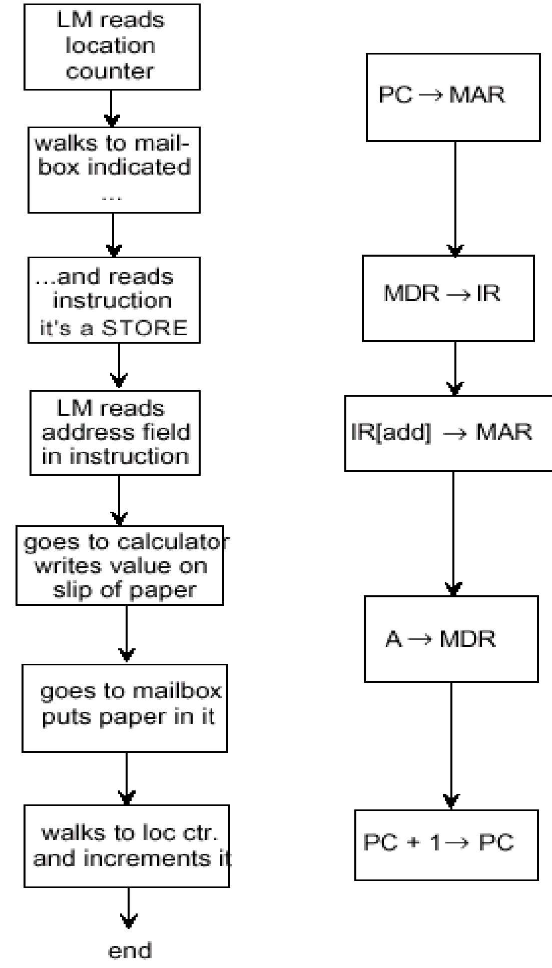
Fetch-Execute Cycle

- Two-cycle process because both instructions and data are in memory
- *Fetch*
  - Decode or find instruction, load from memory into register and signal ALU
- *Execute*
  - Performs operation that instruction requires
  - Move/transform data

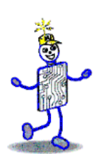
Copyright 2010 John Wiley & Sons, Inc.



LMC vs. CPU
Fetch and Execute Cycle



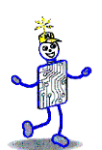
Copyright 2010 John Wiley & Sons, Inc.



Load Fetch/Execute Cycle

1. PC → MAR	Transfer the address from the PC to the MAR
2. MDR → IR	Transfer the instruction to the IR
3. IR[address] → MAR	Address portion of the instruction loaded in MAR
4. MDR → A	Actual data copied into the accumulator
5. PC + 1 → PC	Program Counter incremented

Copyright 2010 John Wiley & Sons, Inc.

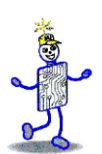


Store Fetch/Execute Cycle

1. PC → MAR	Transfer the address from the PC to the MAR
2. MDR → IR	Transfer the instruction to the IR
3. IR[address] → MAR	Address portion of the instruction loaded in MAR
4. A → MDR*	Accumulator copies data into MDR
5. PC + 1 → PC	Program Counter incremented

*Notice how Step #4 differs for LOAD and STORE

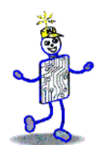
Copyright 2010 John Wiley & Sons, Inc.



ADD Fetch/Execute Cycle

1. PC → MAR	Transfer the address from the PC to the MAR
2. MDR → IR	Transfer the instruction to the IR
3. IR[address] → MAR	Address portion of the instruction loaded in MAR
4. A + MDR → A	Contents of MDR added to contents of accumulator
5. PC + 1 → PC	Program Counter incremented

Copyright 2010 John Wiley & Sons, Inc.



LMC Fetch/Execute

SUBTRACT

PC → MAR

MDR → IR

IR[addr] → MAR

A – MDR → A

PC + 1 → PC

IN

PC → MAR

MDR → IR

IOR → A

PC + 1 → PC

OUT

PC → MAR

MDR → IR

A → IOR

PC + 1 → PC

HALT

PC → MAR

MDR → IR

BRANCH

PC → MAR

MDR → IR

IR[addr] → PC

BRANCH on Condition

PC → MAR

MDR → IR

If condition false: PC + 1 → PC

If condition true: IR[addr] → PC

Copyright 2010 John Wiley & Sons, Inc.

Bus

- The physical connection that makes it possible to transfer data from one location in the computer system to another
- Group of electrical or optical conductors for carrying signals from one location to another
  - Wires or conductors printed on a circuit board
  - *Line*: each conductor in the bus
- 4 kinds of signals
  1. Data
  2. Addressing
  3. Control signals
  4. Power (sometimes)

Copyright 2010 John Wiley & Sons, Inc.

Bus Characteristics

- Number of separate conductors
- Data width in bits carried simultaneously
- Addressing capacity
- Lines on the bus are for a single type of signal or shared
- Throughput - data transfer rate in bits per second
- Distance between two endpoints
- Number and type of attachments supported
- Type of control required
- Defined purpose
- Features and capabilities

Copyright 2010 John Wiley & Sons, Inc.

Bus Categorizations

- Parallel vs. serial buses
- Direction of transmission
  - Simplex – unidirectional
  - Half duplex – bidirectional, one direction at a time
  - Full duplex – bidirectional simultaneously
- Method of interconnection
  - Point-to-point – single source to single destination
    - Cables – point-to-point buses that connect to an external device
  - Multipoint bus – also broadcast bus or multidrop bus
    - Connect multiple points to one another

Copyright 2010 John Wiley & Sons, Inc.

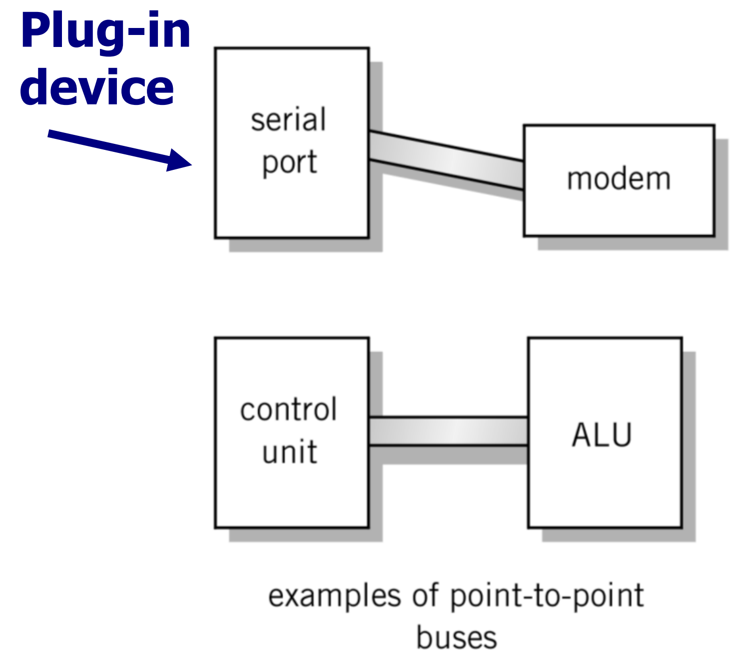
Parallel vs. Serial Buses

- Parallel
  - High throughput because all bits of a word are transmitted simultaneously
  - Expensive and require a lot of space
  - Subject to radio-generated electrical interference which limits their speed and length
  - Generally used for short distances such as CPU buses and on computer motherboards
- Serial
  - 1 bit transmitted at a timed
  - Single data line pair and a few control lines
  - For many applications, throughput is higher than for parallel because of the lack of electrical interference

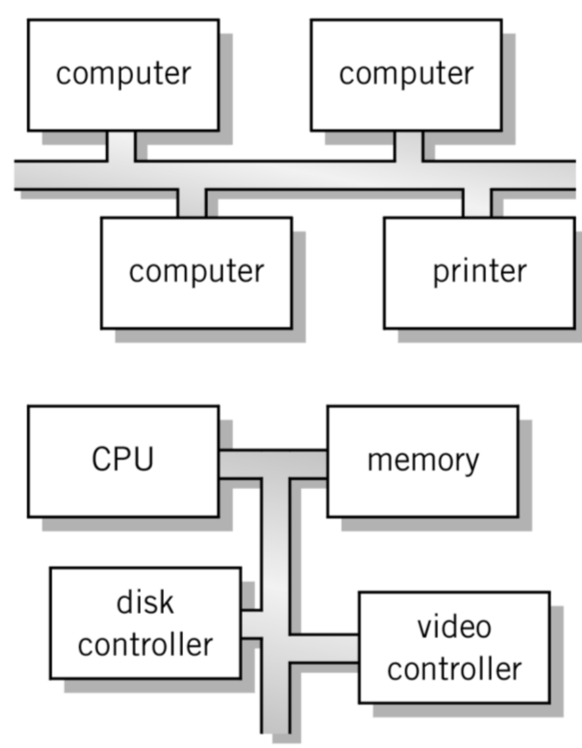
Copyright 2010 John Wiley & Sons, Inc.

Point-to-point vs. Multipoint

Plug-in device



examples of point-to-point buses

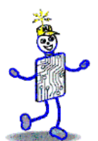


examples of multipoint buses

Broadcast bus Example: Ethernet

Shared among multiple devices

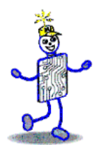
Copyright 2010 John Wiley & Sons, Inc.



Classification of Instructions

- Data Movement (load, store)
  - Most common, greatest flexibility
  - Involve memory and registers
  - What's this size of a *word* ? 16? 32? 64 bits?
- Arithmetic
  - Operators **+ - / * ^**
  - Integers and floating point
- Boolean Logic
  - Often includes at least **AND, XOR,** and **NOT**
- Single operand manipulation instructions
  - Negating, decrementing, incrementing, set to 0

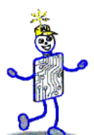
Copyright 2010 John Wiley & Sons, Inc.



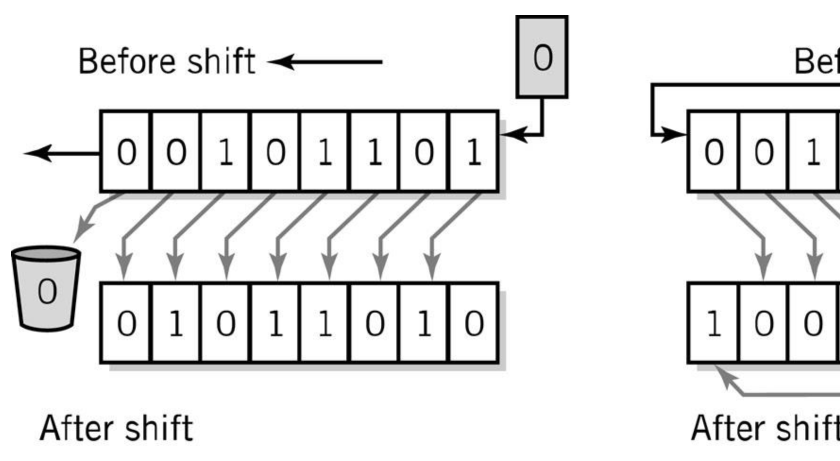
More Instruction Classifications

- Bit manipulation instructions
  - Flags to test for conditions
- Shift and rotate
- Program control
- Stack instructions
- Multiple data instructions
- I/O and machine control

Copyright 2010 John Wiley & Sons, Inc.

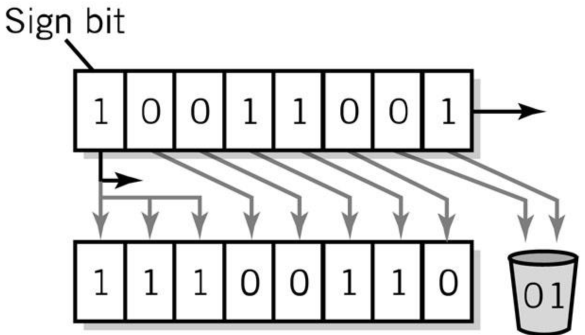


Register Shifts and Rotates



a. Left logical shift register 1 bit

b. Rotate right 1 bit

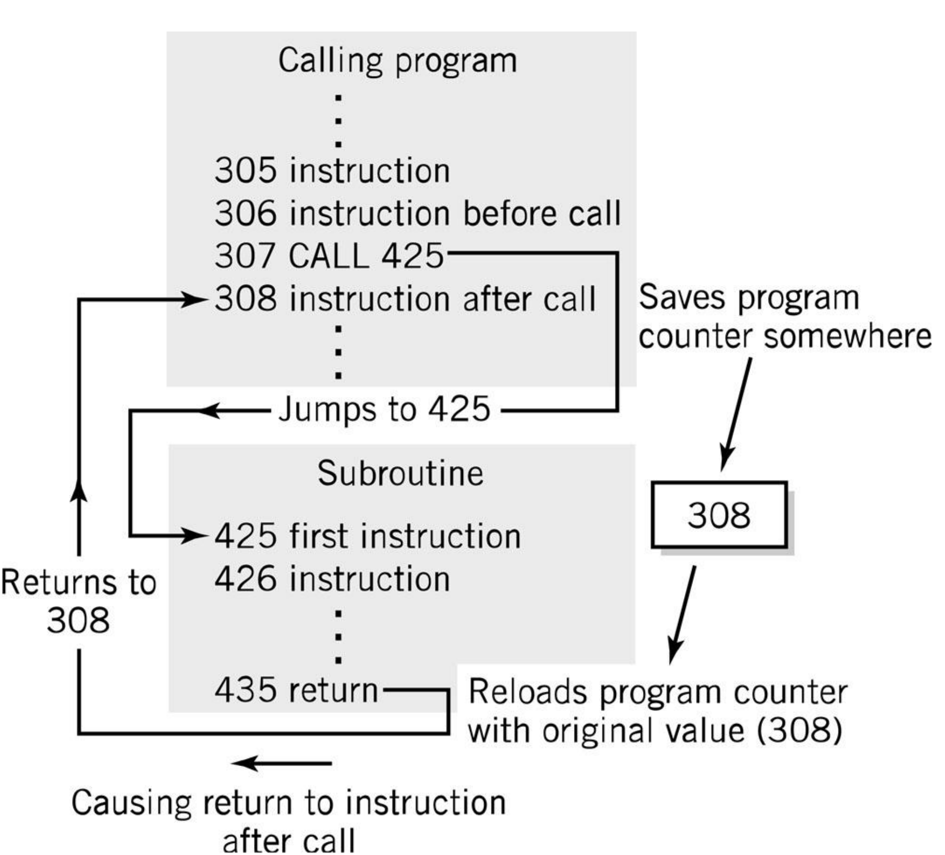


c. Right arithmetic shift 2 bits

Copyright 2010 John Wiley & Sons, Inc.

Program Control Instructions

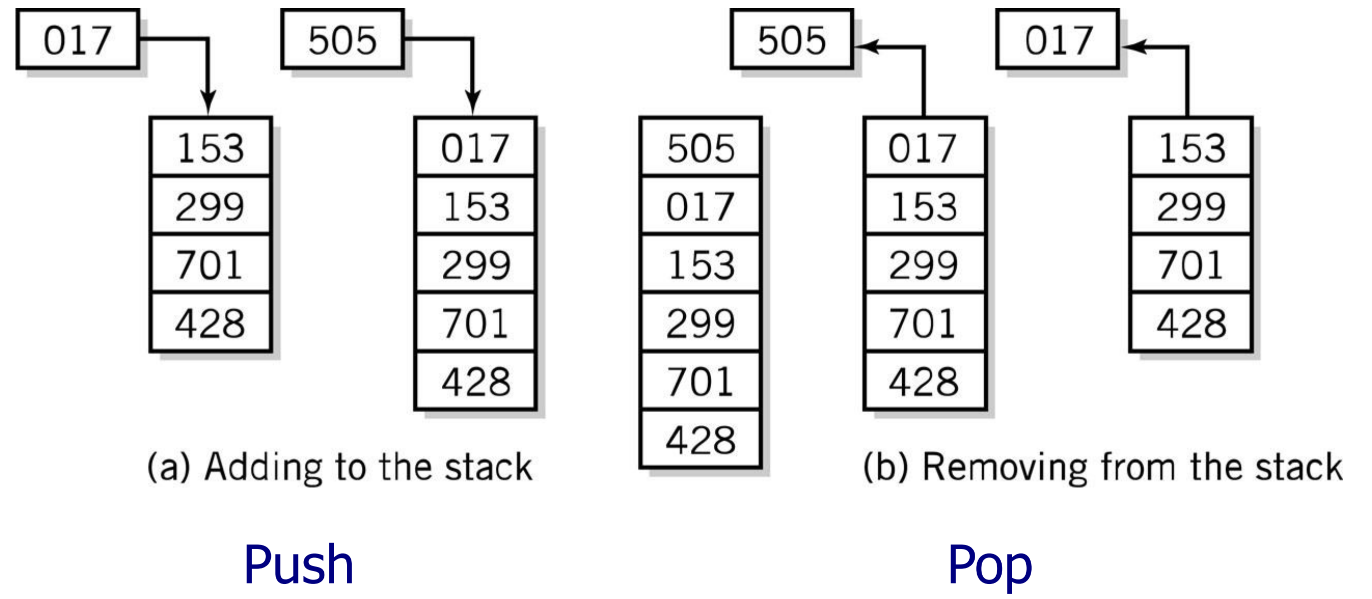
- Program control
  - Jump and branch
  - Subroutine call and return



Copyright 2010 John Wiley & Sons, Inc.

Stack Instructions

- Stack instructions
  - LIFO method for organizing information
  - Items removed in the reverse order from that in which they are added



(a) Adding to the stack

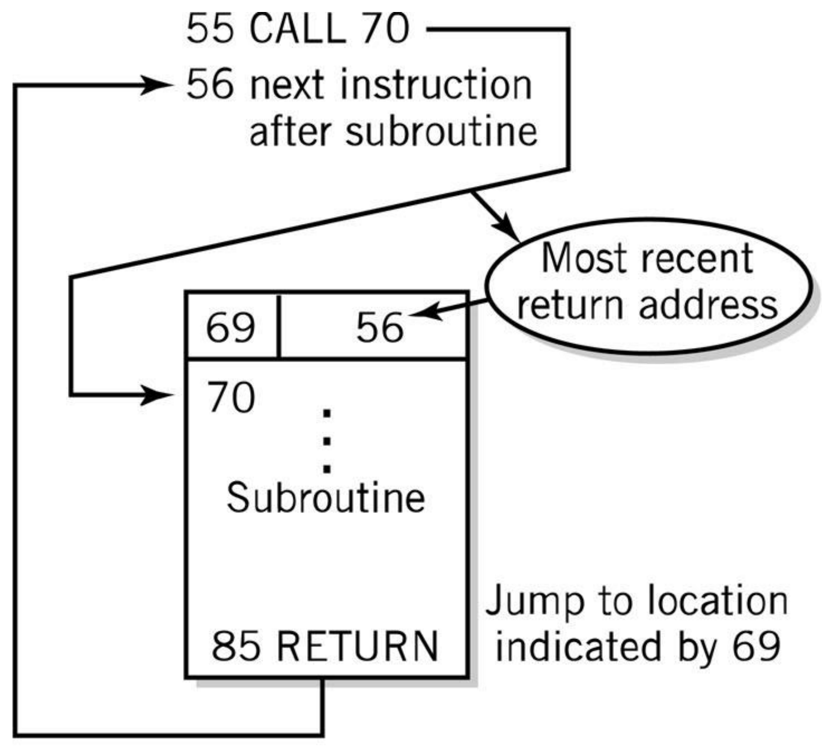
(b) Removing from the stack

Push

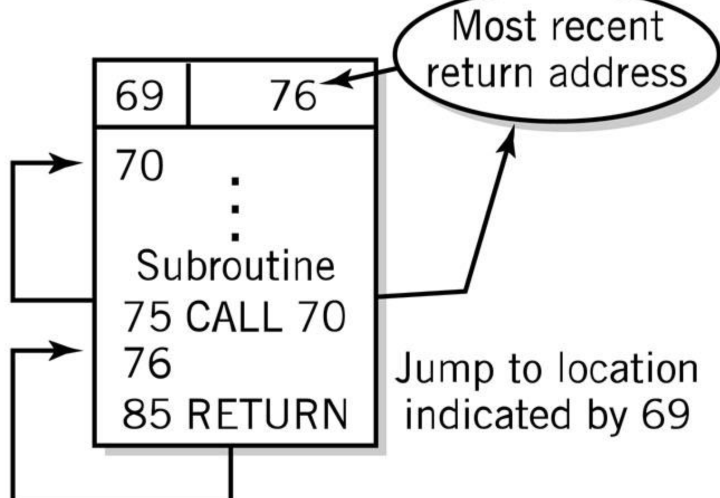
Pop

Copyright 2010 John Wiley & Sons, Inc.

Fixed Location Subroutine Return Address Storage: *Oops!*



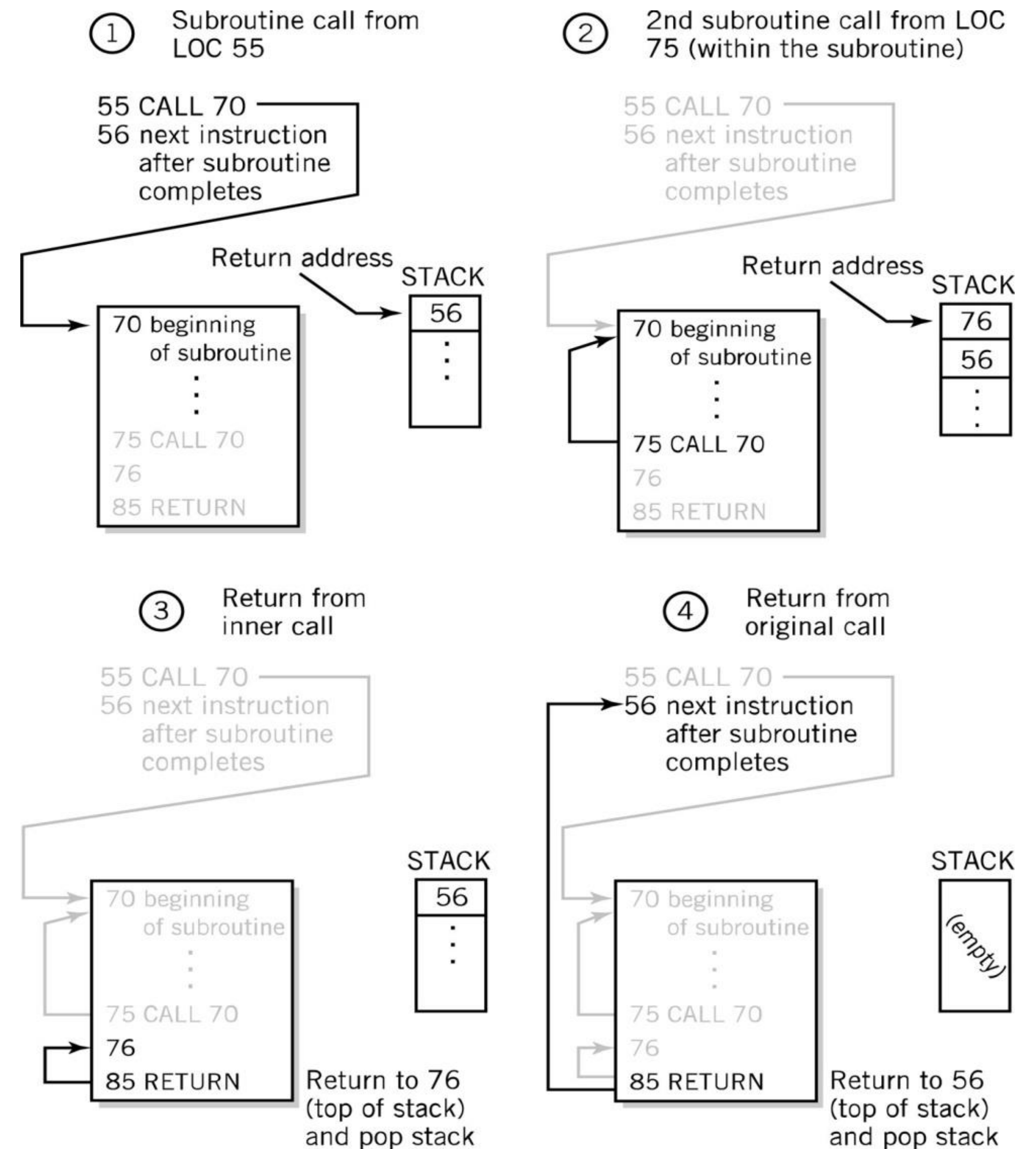
a. Subroutine called from loc.55



b. Subroutine re-called from 75, within the subroutine

Copyright 2010 John Wiley & Sons, Inc.

Stack Subroutine Return Address Storage



Copyright 2010 John Wiley & Sons, Inc.

Block of Memory as a Stack

PUSH increments pointer, then STORES data

323

54

53

Stack pointer

54	323
53	555
52	108
51	919
50	277

Bottom of stack

POP loads data, then decrements pointer

323

54

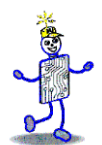
53

Stack pointer

54	323
53	555
52	108
51	919
50	277

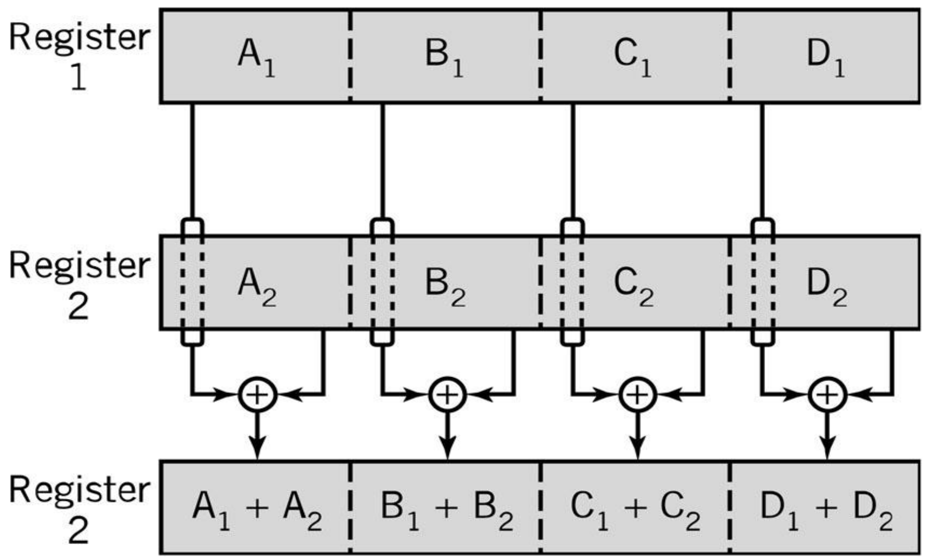
Bottom of stack

Copyright 2010 John Wiley & Sons, Inc.



Multiple Data Instructions

- Perform a single operation on multiple pieces of data simultaneously
  - SIMD: Single Instruction, Multiple Data
  - Commonly used in multimedia, *vector* and array processing applications



Copyright 2010 John Wiley & Sons, Inc.

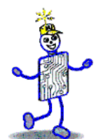
Instruction Elements

- OPCODE: task
- Source OPERAND(s)
- Result OPERAND
  - Location of data (register, memory)
    - Explicit: included in instruction
    - Implicit: default assumed

} Addresses

OPCODE	Source OPERAND	Result OPERAND

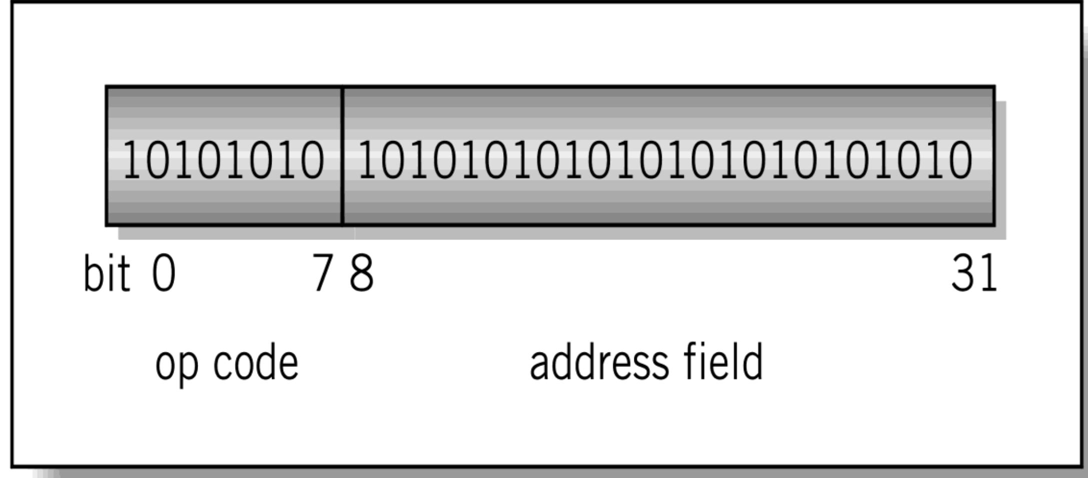
Copyright 2010 John Wiley & Sons, Inc.



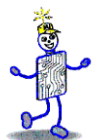
Instruction Format

- *Machine-specific* template that specifies
  - Length of the op code
  - Number of operands
  - Length of operands

Simple 32-bit Instruction Format



Copyright 2010 John Wiley & Sons, Inc.



Instructions

- Instruction
  - Direction given to a computer
  - Causes electrical or optical signals to be sent through specific circuits for processing
- Instruction set
  - Design defines functions performed by the processor
  - Differentiates computer architecture by the
    - Number of instructions
    - Complexity of operations performed by individual instructions
    - Data types supported
    - Format (layout, fixed vs. variable length)
    - Use of registers
    - Addressing (size, modes)

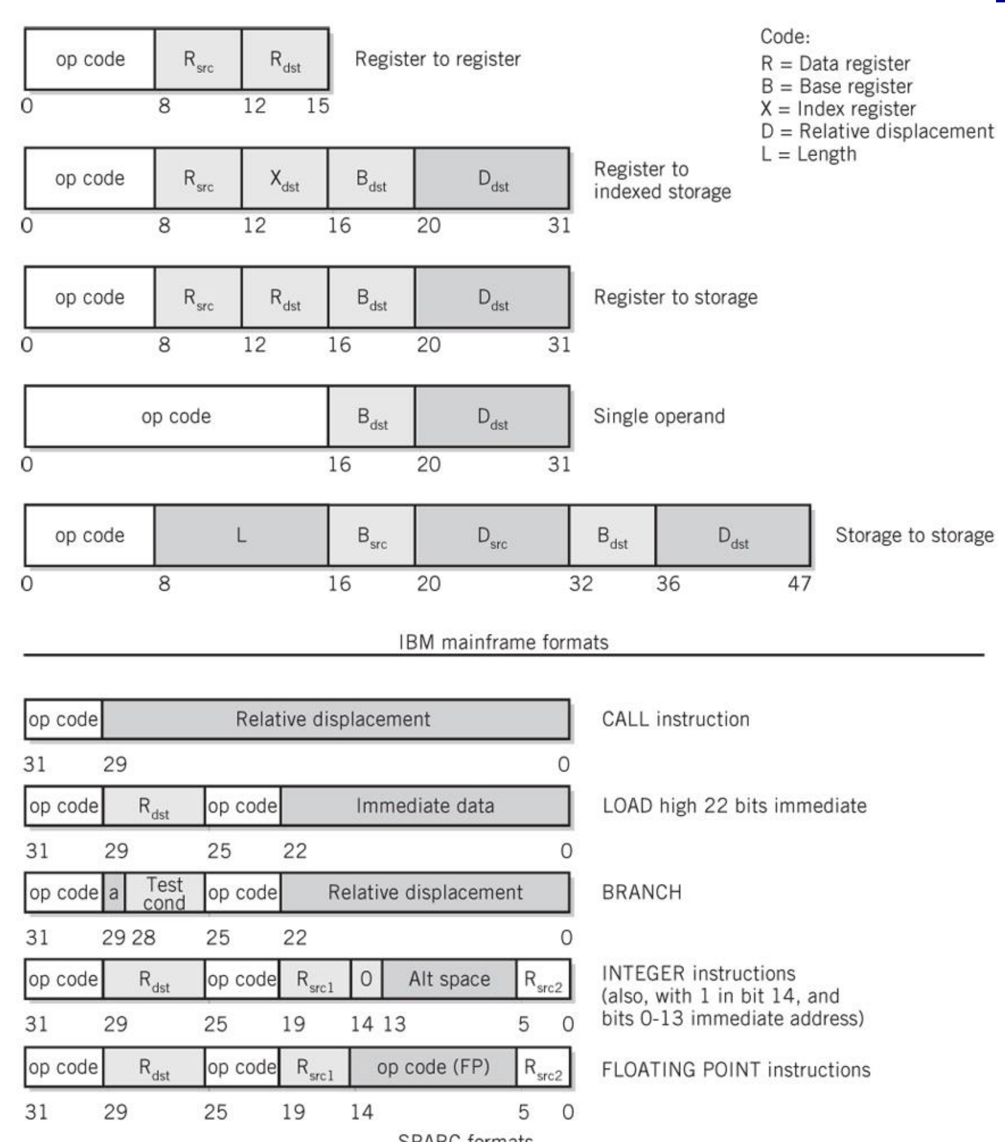
Copyright 2010 John Wiley & Sons, Inc.

Instruction Word Size

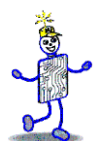
- Fixed vs. variable size
  - Pipelining has mostly eliminated variable instruction size architectures
- Most current architectures use 32-bit or 64-bit words
- Addressing Modes
  - Direct
    - Mode used by the LMC
  - Register Deferred
  - Also immediate, indirect, indexed

Copyright 2010 John Wiley & Sons, Inc.

Instruction Format Examples

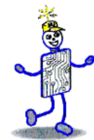


Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 8: CPU and Memory

Design, Enhancement, and Implementation

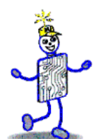
The Architecture of Computer Hardware, Systems Software & Networking:
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley College



Current CPU Architectures

- Current CPU Architecture Designs
  - Traditional modern architectures
  - VLIW (Transmeta) – Very Long Instruction Word
  - EPIC (Intel) – Explicitly Parallel Instruction Computer
- Current CPU Architectures
  - IBM Mainframe series
  - Intel x86 family
  - IBM POWER/PowerPC family
  - Sun SPARC family

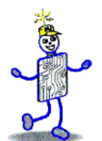
Copyright 2010 John Wiley & Sons, Inc.

Traditional Modern Architectures

Problems with early CPU Architectures and solutions:

- Large number of specialized instructions were rarely used but added hardware complexity and slowed down other instructions
- Slow data memory accesses could be reduced by increasing the number of general purpose registers
- Using general registers to hold addresses could reduce the number of addressing modes and simplify architecture design
- Fixed-length, fixed format instruction words would allow instructions to be fetched and decoded independently and in parallel

Copyright 2010 John Wiley & Sons, Inc.



VLIW Architecture

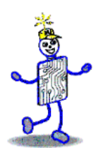
- Transmeta Crusoe CPU
- 128-bit instruction bundle = molecule
  - Four 32-bit atoms (atom = instruction)
  - Parallel processing of 4 instructions
- 64 general purpose registers
- Code morphing layer
  - Translates instructions written for other CPUs into molecules
  - Instructions are not written directly for the Crusoe CPU

Copyright 2010 John Wiley & Sons, Inc.

EPIC Architecture

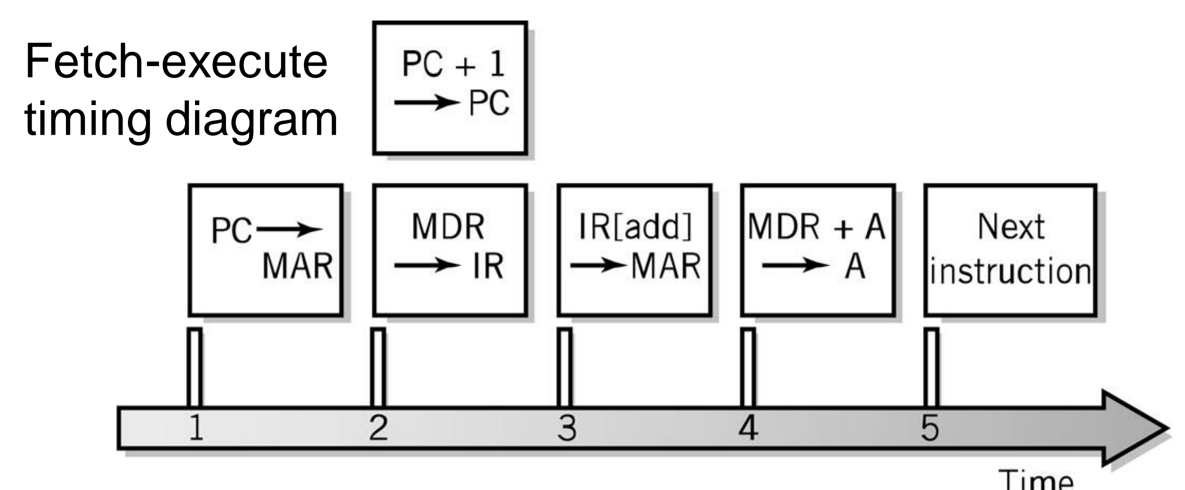
- 128-bit instruction bundle
  - 3 41-bit instructions
  - 5 bits to identify type of instructions in bundle
- 128 64-bit general purpose registers
- 128 82-bit floating point registers
- Intel X86 instruction set included
- Programmers and compilers follow guidelines to ensure parallel execution of instructions

Copyright 2010 John Wiley & Sons, Inc.

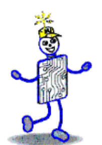


Fetch-Execute Cycle Timing Issues

- Computer clock is used for timing purposes for each step of the instruction cycle
- GHz (gighertz) – billion steps per second
- Instructions can (and often) take more than one step
- Data word width can require multiple steps



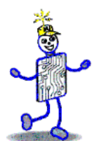
Copyright 2010 John Wiley & Sons, Inc.



CPU Features and Enhancements

- Separate Fetch/Execute Units
- Pipelining
- Multiple, Parallel Execution Units
- Scalar Processing
- Superscalar Processing
- Branch Instruction Processing

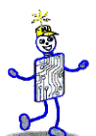
Copyright 2010 John Wiley & Sons, Inc.



Separate Fetch-Execute Units

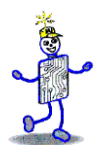
- Fetch Unit
  - Instruction fetch unit
  - Instruction decode unit
    - Determine opcode
    - Identify type of instruction and operands
  - Several instructions are fetched in parallel and held in a buffer until decoded and executed
  - IP – Instruction Pointer register holds instruction location of current being processed
- Execute Unit
  - Receives instructions from the decode unit
  - Appropriate execution unit services the instruction

Copyright 2010 John Wiley & Sons, Inc.



Alternative CPU Organization

Copyright 2010 John Wiley & Sons, Inc.



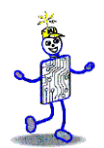
Instruction Pipelining

- Assembly-line technique to allow overlapping between fetch-execute cycles of sequences of instructions
- Scalar processing
  - Average instruction execution is approximately equal to the clock speed of the CPU
- Problems from stalling
  - Instructions have different numbers of steps
- Problems from branching

Copyright 2010 John Wiley & Sons, Inc.

Pipelining Example

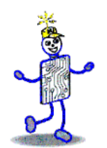
Copyright 2010 John Wiley & Sons, Inc.



Branch Problem Solutions

- Separate pipelines for both possibilities
- Probabilistic approach
- Requiring the following instruction to not be dependent on the branch
- Instruction Reordering (superscalar processing)

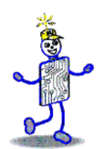
Copyright 2010 John Wiley & Sons, Inc.



Multiple, Parallel Execution Units

- Different instructions have different numbers of steps in their cycle
- Differences in each step
- Each execution unit is optimized for one general type of instruction
- Multiple execution units permit simultaneous execution of several instructions

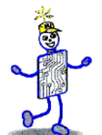
Copyright 2010 John Wiley & Sons, Inc.



Superscalar Processing

- Process more than one instruction per clock cycle
- Separate fetch and execute cycles as much as possible
- Buffers for fetch and decode phases
- Parallel execution units

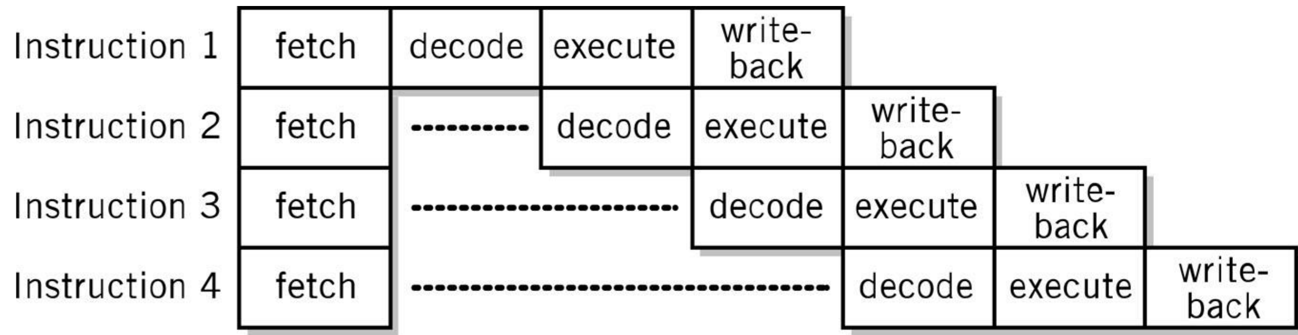
Copyright 2010 John Wiley & Sons, Inc.



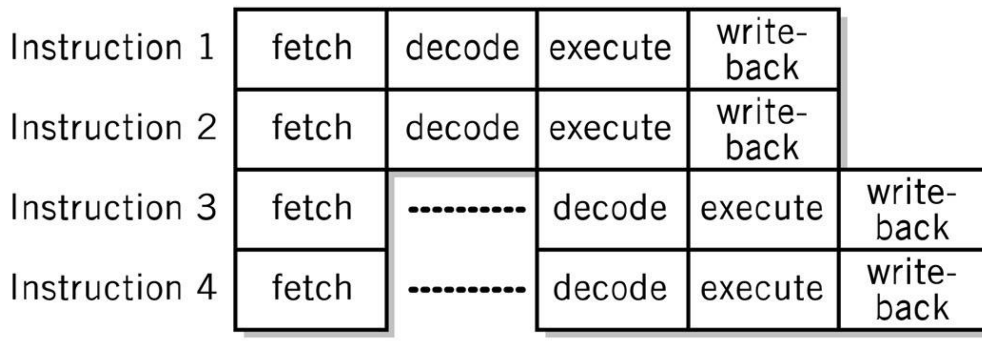
Superscalar CPU Block Diagram

Copyright 2010 John Wiley & Sons, Inc.

Scalar vs. Superscalar Processing

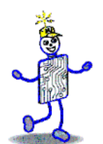


a. Scalar



b. Superscalar

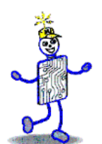
Copyright 2010 John Wiley & Sons, Inc.



Superscalar Issues

- Out-of-order processing – dependencies (hazards)
- Data dependencies
- Branch (flow) dependencies and speculative execution
- Parallel speculative execution or branch prediction
- Branch History Table
- Register access conflicts
  - Rename or logical registers

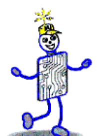
Copyright 2010 John Wiley & Sons, Inc.



Memory Enhancements

- Memory is slow compared to CPU processing speeds!
  - 2Ghz CPU = 1 cycle in ½ of a billionth of a second
  - 70ns DRAM = 1 access in 70 millionth of a second
- Methods to improvement memory accesses
  - Wide Path Memory Access
    - Retrieve multiple bytes instead of 1 byte at a time
  - Memory Interleaving
    - Partition memory into subsections, each with its own address register and data register
  - Cache Memory

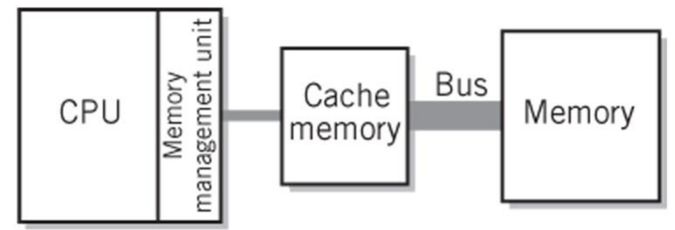
Copyright 2010 John Wiley & Sons, Inc.



Memory Interleaving

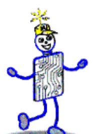
Copyright 2010 John Wiley & Sons, Inc.

Cache Memory



- Blocks: 8 or 16 bytes
- Tags: pointer to location in main memory
- Cache controller
  - hardware that checks tags
- Cache Line
  - Unit of transfer between storage and cache memory
- Hit Ratio: ratio of hits out of total requests
- Synchronizing cache and memory
  - Write through
  - Write back

Copyright 2010 John Wiley & Sons, Inc.



Step-by-Step Use of Cache

CPU

1. Every memory request goes to the cache controller

2334

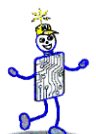
cache controller

2. which checks the request against each tag. (In this illustration, each line contains 4 bytes starting with the tag address.)

tags	data
1032	
2332	
0040	

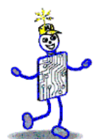
3. If there is a hit, the cache location is used instead of memory.

Copyright 2010 John Wiley & Sons, Inc.



Step-by-Step Use of Cache

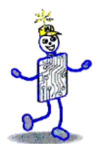
Copyright 2010 John Wiley & Sons, Inc.



Performance Advantages

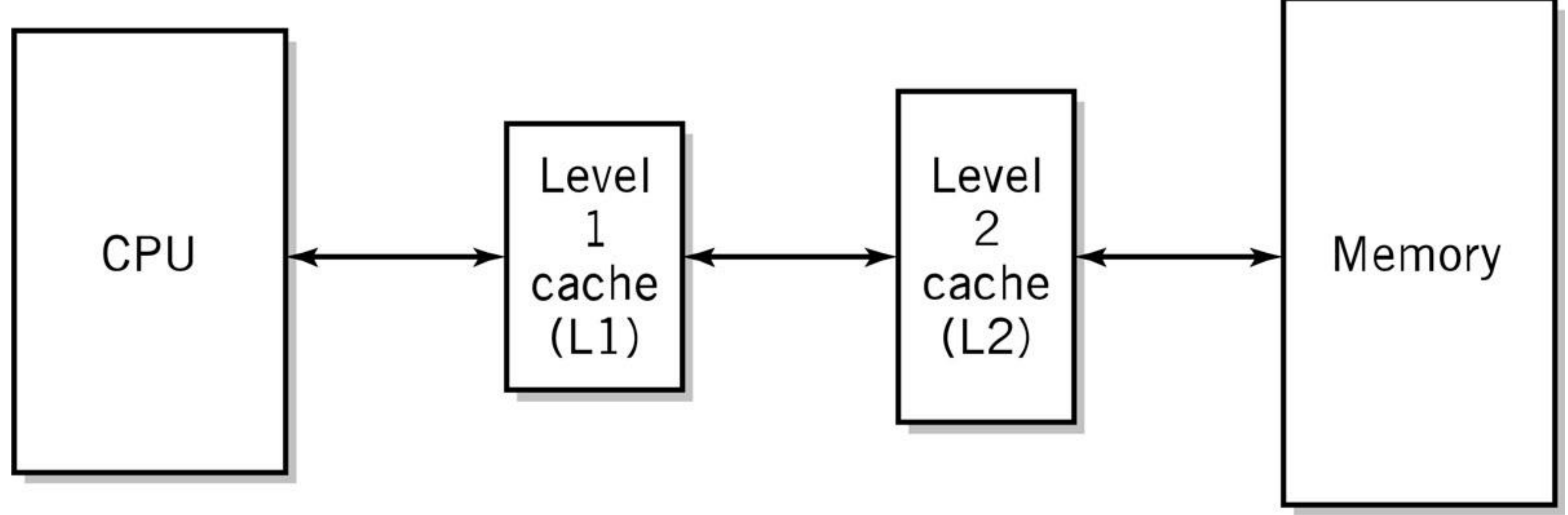
- Hit ratios of 90% common
- 50%+ improved execution speed
- Locality of reference is why caching works
  - Most memory references confined to small region of memory at any given time
  - Well-written program in small loop, procedure or function
  - Data likely in array
  - Variables stored together

Copyright 2010 John Wiley & Sons, Inc.



Two-level Caches

- Why do the sizes of the caches have to be different?



Copyright 2010 John Wiley & Sons, Inc.

Modern CPU Block Diagram

Bus interface unit

Cache memory and memory management unit

Instruction unit

Branch processing unit

Integer processing unit(s)

General-purpose registers

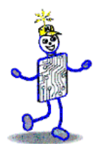
Load/ store unit

Floating point registers

Floating point processing unit(s)

Completion or retire unit

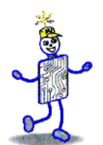
Copyright 2010 John Wiley & Sons, Inc.



Multiprocessing

- Reasons
  - Increase the processing power of a system
  - Parallel processing
- Multiprocessor system
  - Tightly coupled
  - Multicore processors - when CPUs are on a single integrated circuit

Copyright 2010 John Wiley & Sons, Inc.



Multiprocessor Systems

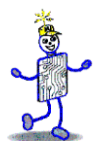
- Identical access to programs, data, shared memory, I/O, etc.
- Easily extends multi-tasking, and redundant program execution
- Two ways to configure
  - Master-slave multiprocessing
  - Symmetrical multiprocessing (SMP)

Copyright 2010 John Wiley & Sons, Inc.

Typical Multiprocessing System Configuration

CPU 1
CPU 2
CPU 3
System bus
Memory
Host/PCI bridge
PCI bus
I/O controller
Other I/O controllers
Disks

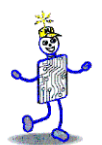
Copyright 2010 John Wiley & Sons, Inc.



Master-Slave Multiprocessing

- Master CPU
  - Manages the system
  - Controls all resources and scheduling
  - Assigns tasks to slave CPUs
- Advantages
  - Simplicity
  - Protection of system and data
- Disadvantages
  - Master CPU becomes a bottleneck
  - Reliability issues – if master CPU fails entire system fails

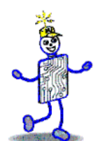
Copyright 2010 John Wiley & Sons, Inc.



Symmetrical Multiprocessing

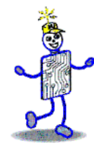
- Each CPU has equal access to resources
- Each CPU determines what to run using a standard algorithm
- Disadvantages
  - Resource conflicts – memory, i/o, etc.
  - Complex implementation
- Advantages
  - High reliability
  - Fault tolerant support is straightforward
  - Balanced workload

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 9: Input / Output

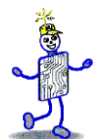
The Architecture of Computer Hardware, Systems Software & Networking:
An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

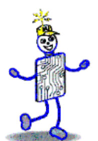
PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley College



Basic Model

- Processing speed or program execution
  - determined primarily by ability of I/O operations to stay ahead of processor.

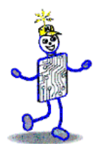
Copyright 2010 John Wiley & Sons, Inc.



I/O Requirements

- Means for addressing different peripheral devices
- A way for peripheral devices to initiate communication with the CPU
- An efficient means of transferring data directly between I/O and memory for large data transfers since programmed I/O is suitable only for slow devices and individual word transfers
- Buses that interconnect high-speed I/O devices with the computer must support high data transfer rates
- Means for handling devices with extremely different control requirements

Copyright 2010 John Wiley & Sons, Inc.



I/O Interfaces

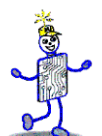
- Are necessary because of
  - Different formats required by the devices
  - Incompatibilities in speed between the devices and the CPU make synchronization difficult
  - Bursts of data vs. streaming data
  - Device control requirements that would tie up too much CPU time

Copyright 2010 John Wiley & Sons, Inc.

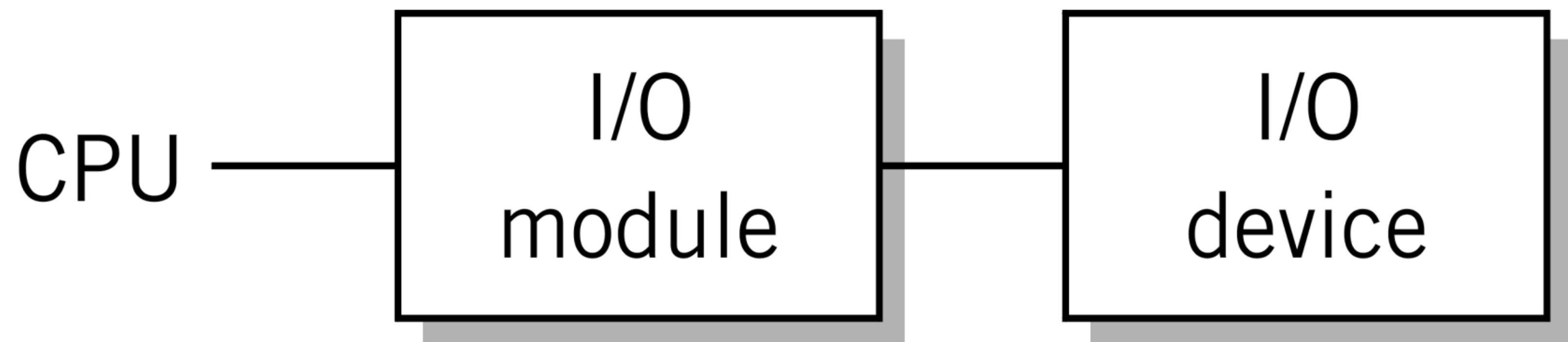
Examples of I/O Devices

Device	Input/Output	Data rate	Type
Keyboard	Input	100 bps	char
Mouse	Input	3800 bps	char
Voice input/output	Input/Output	264 Kbps	block burst
Sound input	Input	3 Mbps	block burst or steady
Scanner	Input	3.2 Mbps	block burst
Laser printer	Output	3.2 Mbps	block burst
Sound output	Output	8 Mbps	block burst or steady
Flash drive	Storage	480-800 Mbps read; 80 Mbps write	block burst
USB	Input or output	1.6-480 Mbps	block burst
Network/Wireless LAN	Input or output	11-100 Mbps	block burst
Network/LAN	Input or output	100-1000 Mbps	block burst
Graphics display	Output	800-8000 Mbps	block burst or steady
Optical disk	Storage	4-400 Mbps	block burst or steady
Magnetic tape	Storage	32-90 Mbps	block burst or steady
Magnetic disk	Storage	240-3000 Mbps	block burst

Copyright 2010 John Wiley & Sons, Inc.

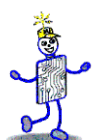


Simple I/O Configuration



Copyright 2010 John Wiley & Sons, Inc.

More Complex I/O Module



Advanced I/O Techniques

- Programmed I/O
  - CPU controlled I/O
- Interrupt Driven I/O
  - External input controls
- Direct Memory Access Controllers
  - Method for transferring data between main memory and a device that bypasses the CPU

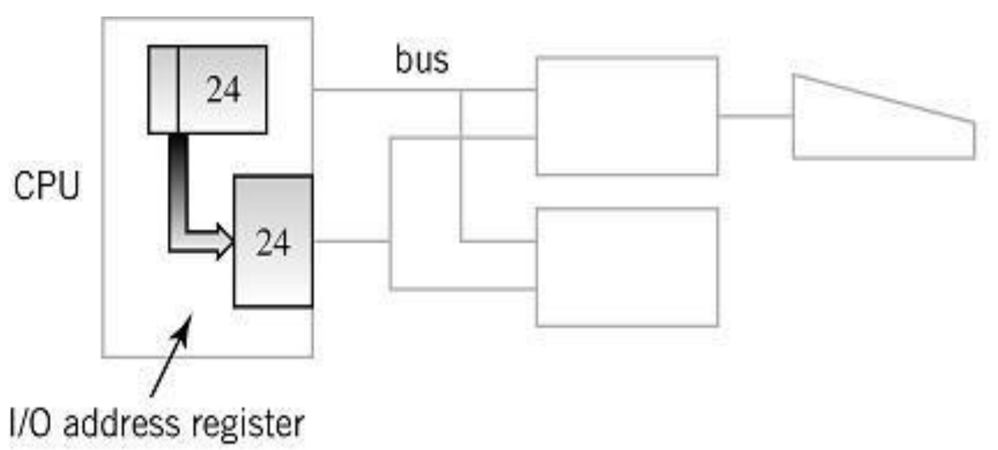
Copyright 2010 John Wiley & Sons, Inc.

Programmed I/O

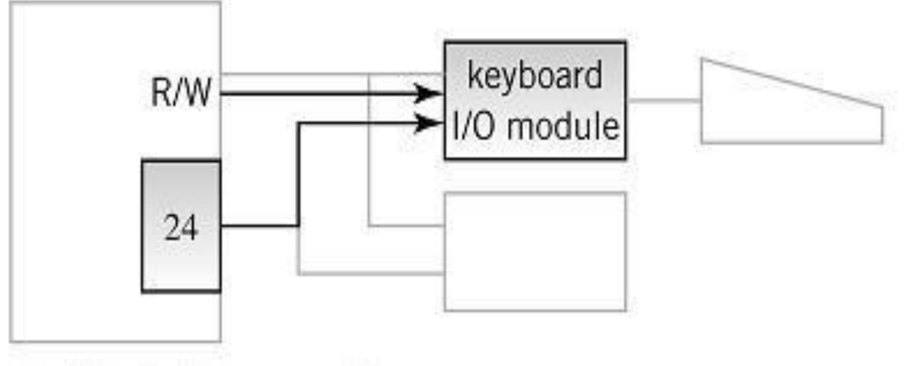
- I/O data and address registers in CPU
- One word transfer per I/O instruction
- Address information for each I/O device
  - LMC I/O capability for 100 devices
- Full instruction fetch/execute cycle
- Primary use:
  - keyboards
  - communication with I/O modules (see DMA)

Copyright 2010 John Wiley & Sons, Inc.

Programmed I/O Example



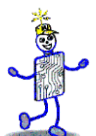
1. CPU executes INPUT 24 instruction. Address 24 is copied to the I/O address register.



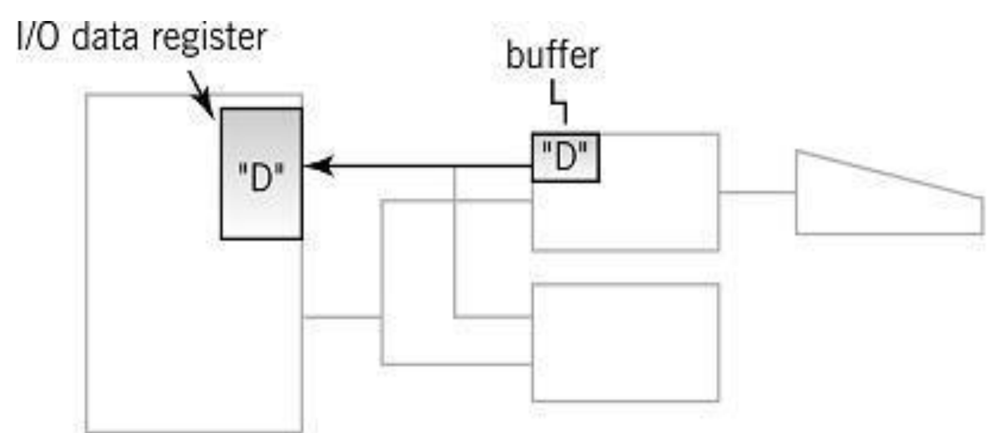
2. Address 24 is recognized by the keyboard I/O module. A read/write control line indicates that the instruction is an INPUT.

(Figure continues on next slide)

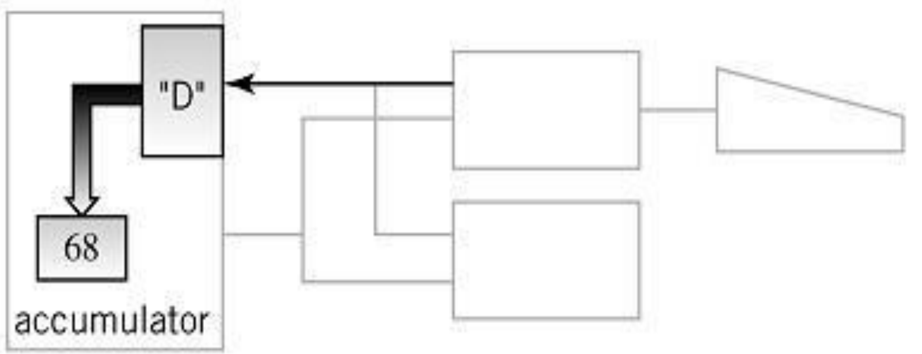
Copyright 2010 John Wiley & Sons, Inc.



Programmed I/O Example



3. A buffer in the I/O module holds a keystroke, in this case ASCII 68, the letter "D". The data is transferred to the I/O data register.



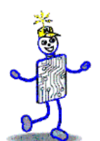
4. From there it is copied to the appropriate accumulator or general-purpose register, completing the operation.

Copyright 2010 John Wiley & Sons, Inc.

Interrupts

- Signal that causes the CPU to alter its normal flow of instruction execution
  - frees CPU from waiting for events
  - provides control for external I/O initiation
- Examples
  - unexpected input
  - abnormal situation
  - illegal instructions
  - multitasking, multiprocessing

Copyright 2010 John Wiley & Sons, Inc.



Interrupt Terminology

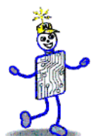
- Interrupt lines (hardware)
  - One or more special control lines to the CPU
- Interrupt request
- Interrupt handlers
  - Program that services the interrupt
  - Also known as an interrupt routine or device driver
- Context
  - Saved registers of a program before control is transferred to the interrupt handler
  - Allows program to resume exactly where it left off when control returns to interrupted program

Copyright 2010 John Wiley & Sons, Inc.

Use of Interrupts

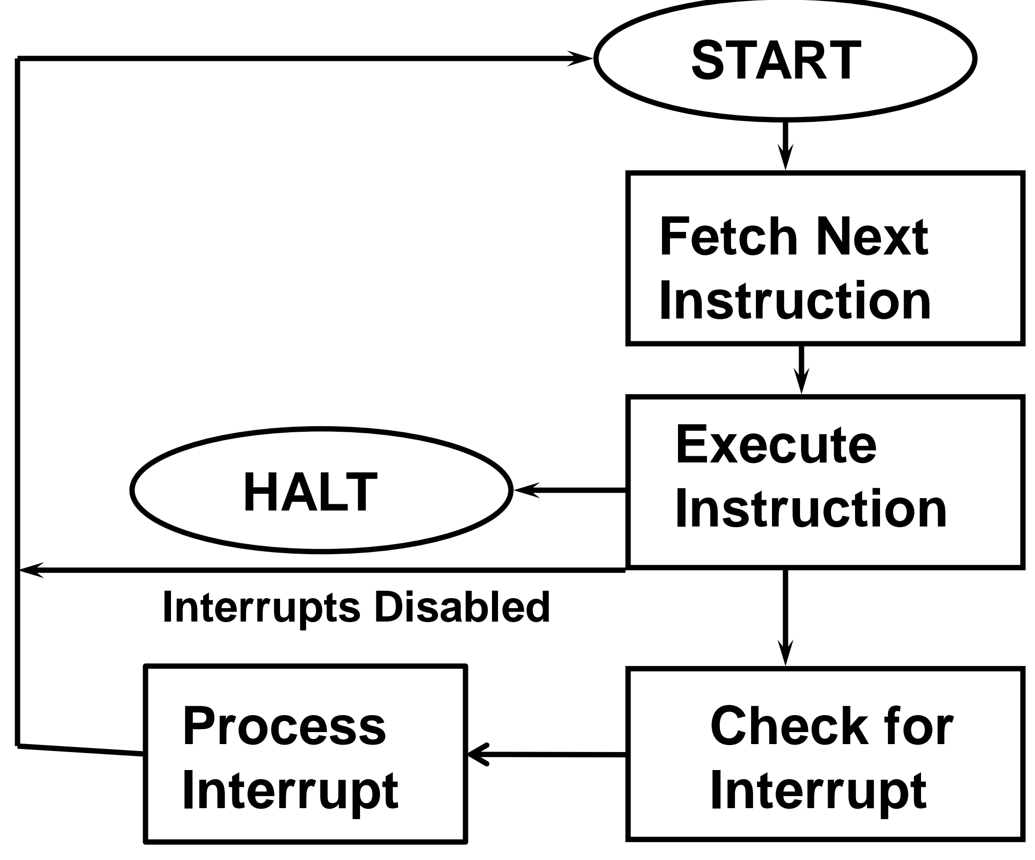
- Notify that an external event has occurred
  - real-time or time-sensitive
- Signal completion
  - printer ready or buffer full
- Allocate CPU time
  - time sharing
- Indicate abnormal event (CPU originates for notification and recovery)
  - illegal operation, hardware error
- Software interrupts

Copyright 2010 John Wiley & Sons, Inc.

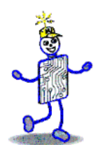


The CPU - The Interrupt Cycle

- Fetch / Execute cycle
- Interrupt cycle



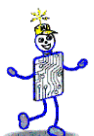
Copyright 2010 John Wiley & Sons, Inc.



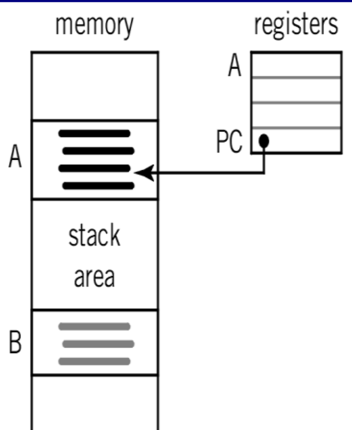
Servicing the Interrupt

1. Lower priority interrupts are held until higher priority interrupts are complete
2. Suspend program in progress
3. Save context, including last instruction executed and data values in registers, in the PCB or the stack area in memory
4. Branch to interrupt handler program

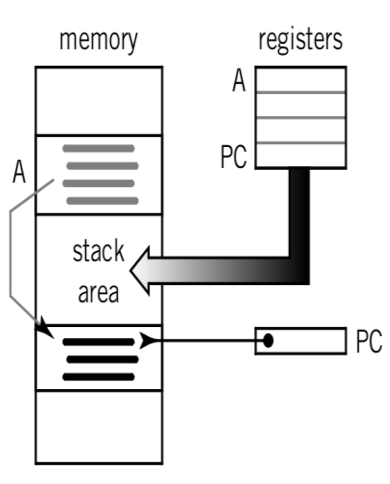
Copyright 2010 John Wiley & Sons, Inc.



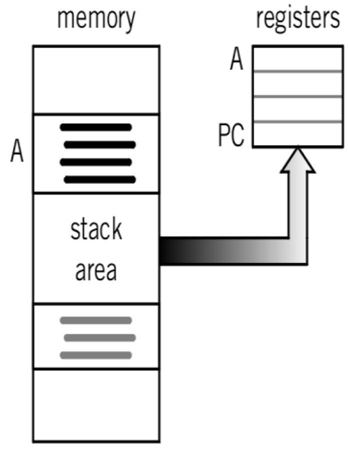
Servicing an Interrupt



1. Before interrupt arrives, program A is executing. The program counter points to the current instruction.

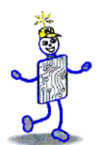


2. When the interrupt is received by the CPU, the current instruction is completed, all the registers are saved in the stack area (or in a special area known as a process control block). The PC is loaded with the starting location of program B, the interrupt handler program. This causes a jump to program B, which becomes the executing program.



3. When the interrupt routine is complete, the registers are restored, including the program counter, and the original program resumes exactly where it left off.

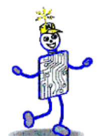
Copyright 2010 John Wiley & Sons, Inc.



Interrupt Processing Methods

- Vectored interrupt
  - Address of interrupting device is included in the interrupt
  - Requires additional hardware to implement
- Polling
  - Identifies interrupting device by polling each device
  - General interrupt is shared by all devices

Copyright 2010 John Wiley & Sons, Inc.



Vectored Interrupts

Interrupt K occurs

Memory

Address of interrupt A

Address of interrupt B

. . .

Address of interrupt K

. . .

Interrupt A service routine

. . .

Interrupt K service routine

Jump to K service routine

Copyright 2010 John Wiley & Sons, Inc.

Polled Interrupts

Interrupt K occurs

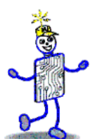
Memory

General interrupt polling routine
. . .
Interrupt A service routine
. . .
Interrupt K service routine

Polls devices to determine which device, then

Jump to K service routine

Copyright 2010 John Wiley & Sons, Inc.



Print Handler Interrupt

Original program executing

Software interrupt to print handler

Suspended

Print interrupt handler

Fill printer buffer

Resume other work

Printer ready interrupt

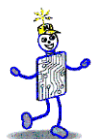
More to print?

Y

N

Continue

Copyright 2010 John Wiley & Sons, Inc.



Using an Interrupt for Time Sharing

Program 1
Program 2
Operating system dispatcher program

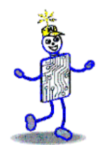
Executing
Time
Clock interrupt
Suspended
Select next program
One quantum
Resume executing program 2
Clock interrupt
Suspended
Select next program
. . .

Copyright 2010 John Wiley & Sons, Inc.

Multiple Interrupts Example

Time	Original program	Interrupt service A	Interrupt service B	Interrupt service C
	Executing			
Interrupt A occurs	Suspended →	Executing		
Interrupt B occurs		Suspended →	Executing	
		Executing	← Completed	
Interrupt C occurs		Suspended →		Executing
		Executing	←	Completed
	Executing	← Completed		

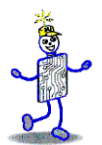
Copyright 2010 John Wiley & Sons, Inc.



Direct Memory Access

- Transferring large blocks of data
- Direct transfer to and from memory
- CPU not actively involved in transfer itself
- Required conditions for DMA
  - The I/O interface and memory must be connected
  - The I/O module must be capable of reading and writing to memory
  - Conflicts between the CPU and the I/O module must be avoided
  - Interrupt required for completion

Copyright 2010 John Wiley & Sons, Inc.



DMA Instructions

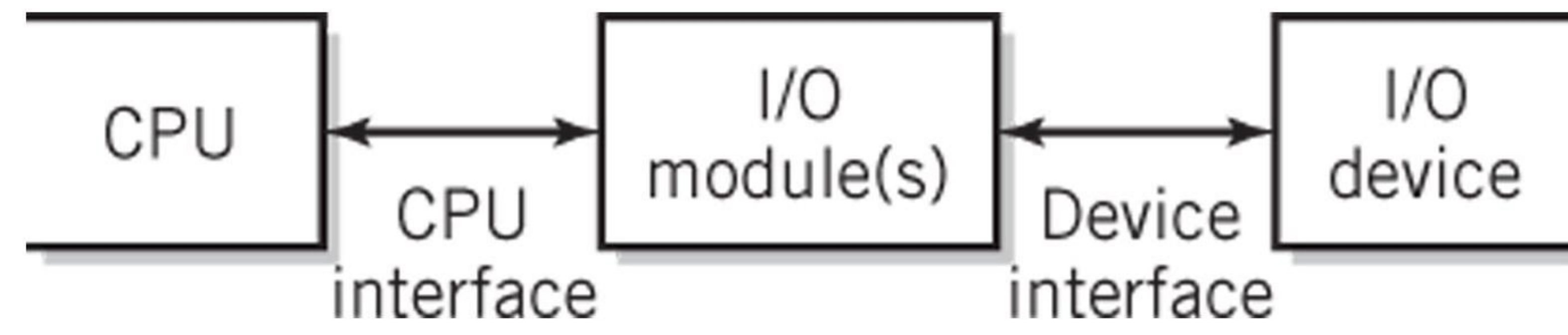
- Application program requests I/O service from operating system
  - privileged programmed I/O instructions
- To initiate DMA, programmed I/O is used to send the following information:
  1. location of data on I/O device
  2. the starting location in memory
  3. the size of the block
  4. read/write
- Interrupt to CPU upon completion of DMA

Copyright 2010 John Wiley & Sons, Inc.

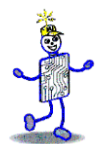
DMA Initiation and Control

Copyright 2010 John Wiley & Sons, Inc.

I/O Module Interfaces



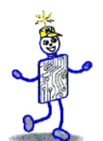
Copyright 2010 John Wiley & Sons, Inc.



I/O Module Functions

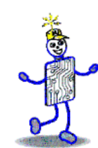
- Recognizes messages from device(s) addressed to it and accepts commands from the CPU
- Provides a buffer where the data from memory can be held until it can be transferred to the device
- Provides the necessary registers and controls to perform a direct memory transfer
- Physically controls the device
- Copies data from its buffer to the device/from the CPU to its buffer
- Communicates with CPU

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."



CHAPTER 10: Computer Peripherals

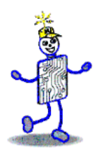
The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach

4th Edition, Irv Englander

John Wiley and Sons ©2010

PowerPoint slides authored by Wilson Wong, Bentley University

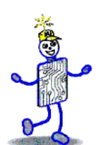
PowerPoint slides for the 3rd edition were co-authored with Lynne Senne, Bentley College



Peripherals

- Devices that are separate from the basic computer
  - Not the CPU, memory, or power supply
- Classified as input, output, and storage
- Connect via
  - Ports
  - Interface to systems bus

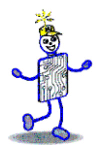
Copyright 2010 John Wiley & Sons, Inc.



Storage Devices

- Primary memory
- Secondary storage
  - Data and programs must be copied to primary memory for CPU access
  - Permanence of data - nonvolatile
  - Direct access storage devices (DASDs)
  - Online storage
  - Offline storage – loaded when needed
  - Network file storage
    - File servers, web servers, database servers

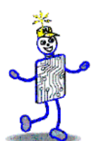
Copyright 2010 John Wiley & Sons, Inc.



Speed

- Measured by access time and data transfer rate
- Access time: average time it takes a computer to locate data and read it
  - millisecond = one-thousandth of a second
- Data transfer rate: amount of data that moves per second

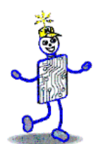
Copyright 2010 John Wiley & Sons, Inc.



Storage Hierarchy

Device	Typical access times
CPU registers	0.25 nsec
Cache memory (SRAM)	1-10 nsec
Conventional memory (DRAM)	10-50 nsec
Flash memory	120 µsec
Magnetic disk drive	10-50 msec
Optical disk drive	100-500 msec
Magnetic tape	0.5 and up sec

Copyright 2010 John Wiley & Sons, Inc.



Secondary Storage Devices

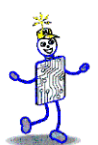
- Solid state memory
- Magnetic disks
- Optical disk storage
- Magnetic tape
- Network storage
- Characteristics
  - Rotation vs. Linear
  - Direct access vs. Sequential access

Copyright 2010 John Wiley & Sons, Inc.

Flash Memory

- Nonvolatile electronic integrated circuit memory
- Similar to other read-only memory but uses a different technology
- Permits reading and writing individual bytes or small blocks of data
- Small size makes it useful in portable devices such as USB “thumb drives”, digital cameras, cell phones, music players
- Relatively immune to physical shocks
- Generates little heat or noise

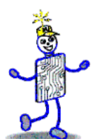
Copyright 2010 John Wiley & Sons, Inc.



Disk Layouts – CAV vs. CLV

- CAV – Constant Angular Velocity
  - Number of bits on each track is the same! Denser towards the center.
  - Spins the same speed for every track
- CLV – Constant Linear Velocity
  - All tracks have the same physical length and number of bits
  - Constant speed reading data off a track
  - Drive has to speed up when accessing close to the center of the drive and slow down when accessing towards the edge of the drive

Copyright 2010 John Wiley & Sons, Inc.



Disk Layout – Multiple Zone

- Multiple zone recording
  - Also known as zone bit recording (ZBR) or zone-CAV recording (Z-CAV)
  - Compromise between CAV and CLV
  - Disk divided into zones
  - Cylinders in different zones have a different number of sectors
  - Number of sectors in a particular zone is constant
  - Data is buffered so the data rate to the I/O interface is constant

Copyright 2010 John Wiley & Sons, Inc.

Multiple-Zone Disk Configuration

Zone 1
Zone 2
Zone 3
Zone 4
Sectors

Copyright 2010 John Wiley & Sons, Inc.

Magnetic Disks

- Track – circle
- Cylinder – same track on all platters
- Block – small arc of a track
- Sector – pie-shaped part of a platter
- Head – reads data off the disk as disk rotates at high speed (4200-14000 RPM)

- Head crash
  - Disk damaged if head touches disk surface
- Parked heads

Copyright 2010 John Wiley & Sons, Inc.

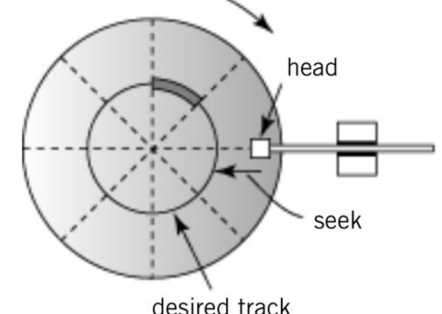
A Hard Disk Layout

Platter
Track
Cylinder
Drive motor

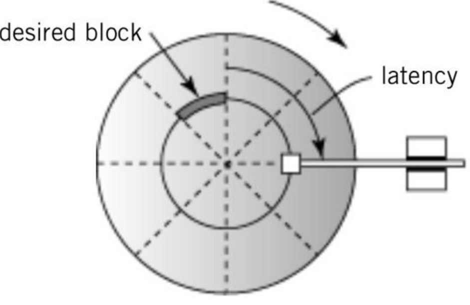
Sector
Block
Head motor
Head, on moving arm
Track

Locating a Block of Data

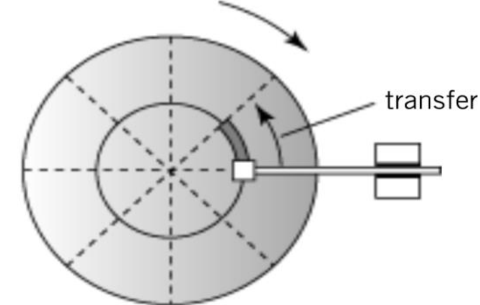
- Average seek time: time required to move from one track to another



- Latency: time required for disk to rotate to beginning of correct sector



- Transfer time: time required to transfer a block of data to the disk controller buffer



Copyright 2010 John Wiley & Sons, Inc.

Disk Access Times

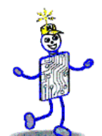
- Average Seek time
  - average time to move from one track to another
- Average Latency time
  - average time to rotate to the beginning of the sector
  - Average Latency time = ½ * 1/rotational speed
- Transfer time
  - 1/(# of sectors * rotational speed)
- Total Time to access a disk block
  - Avg. seek time + avg. latency time + avg. transfer time

Copyright 2010 John Wiley & Sons, Inc.

Magnetic Disks

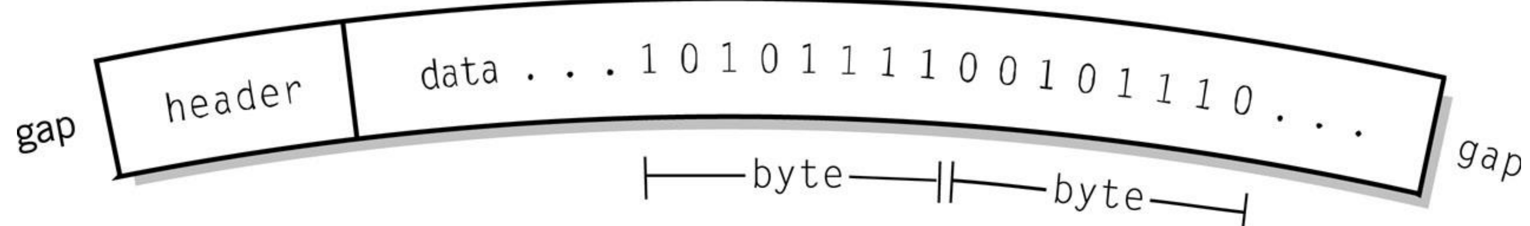
- Data Block Format
  - Interblock gap
  - Header
  - Data
- Formatting disk
  - Establishes the track positions, blocks and headers needed before use of the disk

Copyright 2010 John Wiley & Sons, Inc.

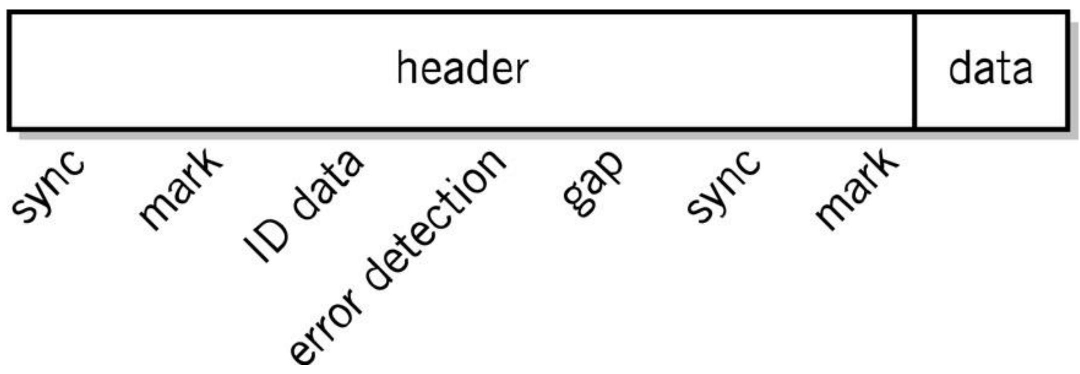


Disk Block Formats

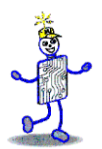
Single Data Block



Header for Windows disk



Copyright 2010 John Wiley & Sons, Inc.



Disk Arrays

- Grouping of multiple disks together
- RAID – Redundant Array of Inexpensive Disks
  - Mirrored array
  - Striped array
  - RAID 0 to RAID 5

Copyright 2010 John Wiley & Sons, Inc.

RAID – Mirrored

- Pair of disks contain the exact same stores of data
- Reading data – alternate blocks of data are read from hard drives and combined
- Access time is reduced by approximately a factor equal to the number of disk drives in array
- Read failure – block is marked and then read from the mirrored drive
- When using three or more mirrored drives, majority logic is used in the event of a failure. Fault-tolerant computers use this technique.

Copyright 2010 John Wiley & Sons, Inc.

RAID - Striped

- A file segment is stored divided into blocks on different disks
- Minimum of three drives needed because one disk drive is reserved for error checking
- Writes – block of parity words from each block of data is created and put on the reserved error checking disk
- Reads – parity data is used to check original data

Copyright 2010 John Wiley & Sons, Inc.

RAID Levels

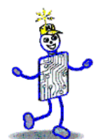
- RAID 0 – not true RAID, no error checking or redundancy, but data is placed across all drives for increased speed
- RAID 1 – mirrored array
- RAID 2, 3, 4 – arrays that are striped in different ways
- RAID 5 – error checking blocks are spread across all drives

Copyright 2010 John Wiley & Sons, Inc.

Optical Storage

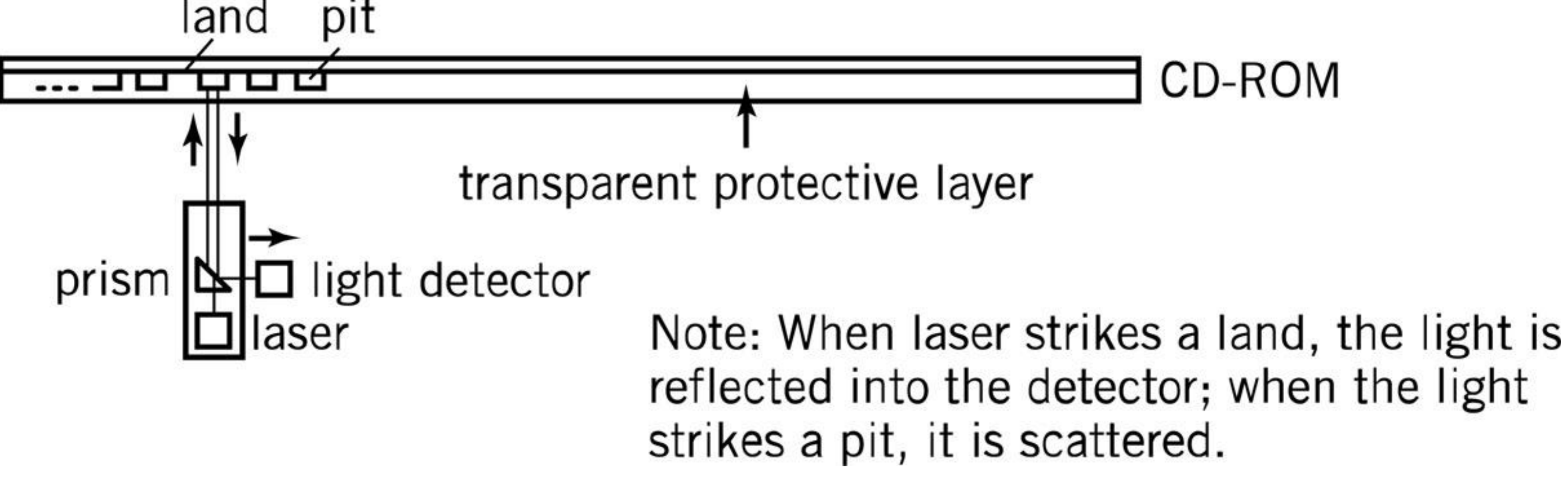
- Reflected light off a mirrored or pitted surface
- CD-ROM
  - 650 MB of data, approximately 550 MB after formatting and error checking
  - Spiral 3 miles long, containing 15 billion bits!
  - CLV – all blocks are same physical length
  - Block – 2352 bytes
    - 2k of data (2048 bytes)
    - 16 bytes for header (12 start, 4 id)
    - 288 bytes for advanced error control
- DVD – similar technology to CD-ROM
- WORM – write-once read-many

Copyright 2010 John Wiley & Sons, Inc.

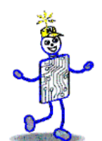


Optical Storage

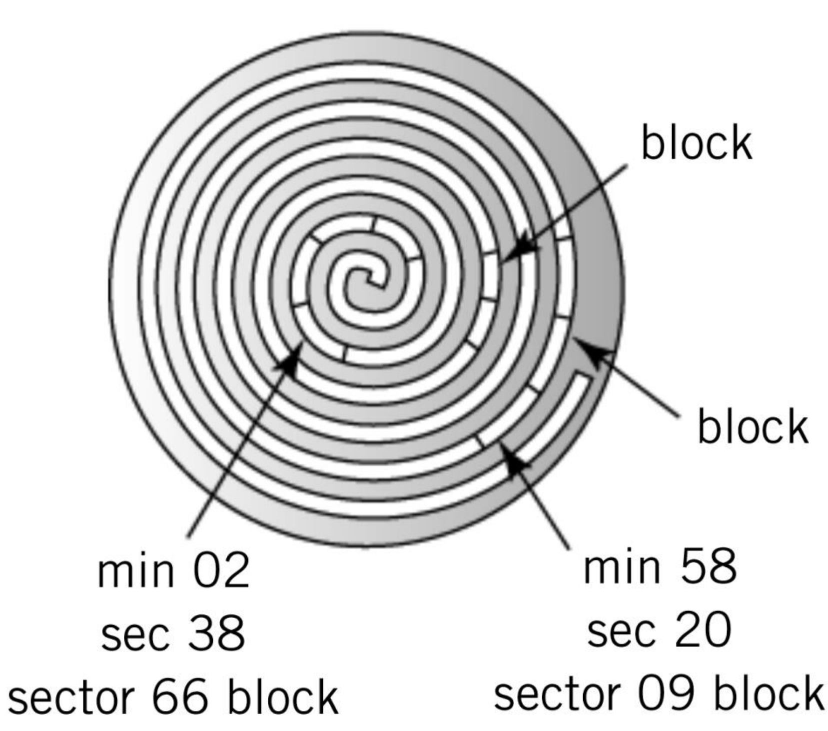
- Laser strikes land: light reflected into detector
- Laser strikes a pit: light scattered

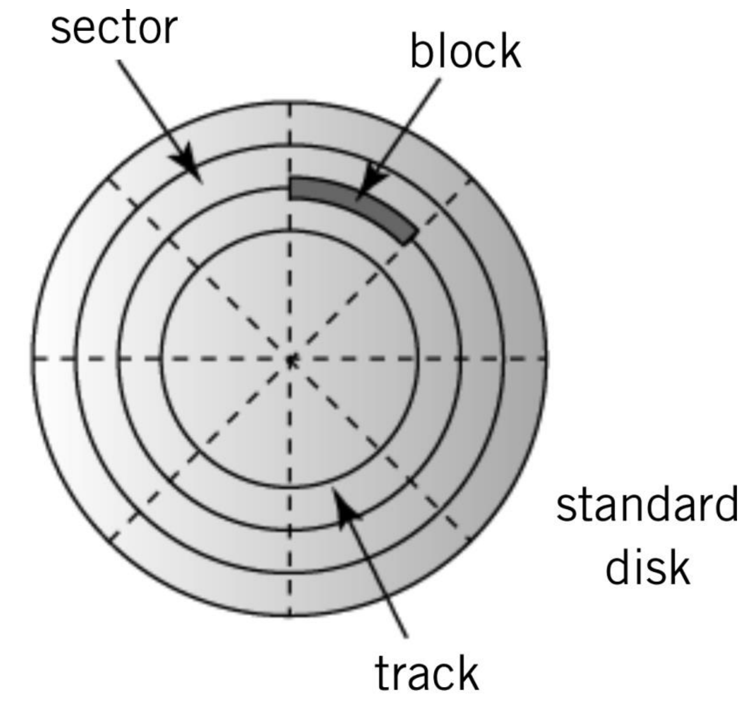


Copyright 2010 John Wiley & Sons, Inc.

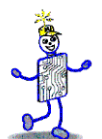


Layout: CD-ROM vs. Standard Disk





Copyright 2010 John Wiley & Sons, Inc.



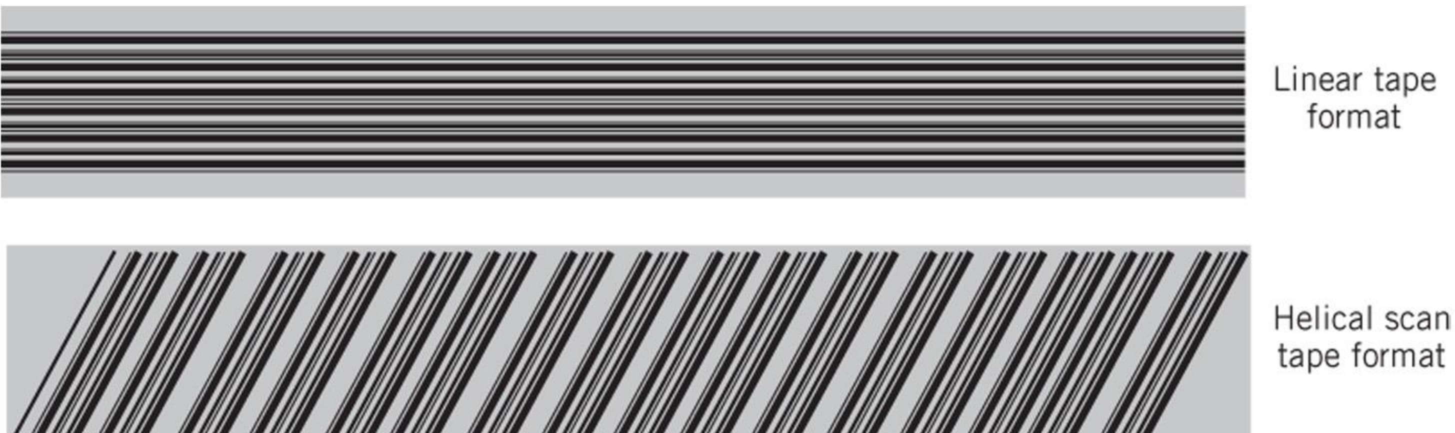
Types of Optical Storage

- WORM Disks
  - Write-once-read-many times
  - Medium can be altered by using a medium-powered laser to blister the surface
- Medium-powered laser blister technology also used for
  - CD-R, DVD-R, DVD-R, DVD+R
  - CD-RW, DVD-RW, DVD+RW, DVD-RAM, DVD+RAMBD-RE
- File compatibility issues between the different CD, DVD and WORM formats

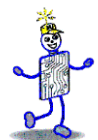
Copyright 2010 John Wiley & Sons, Inc.

Magnetic Tape

- Offline storage
- Archival purposes
- Disaster recovery
- Tape Cartridges
  - Linear tape open format vs. helical scan tape format

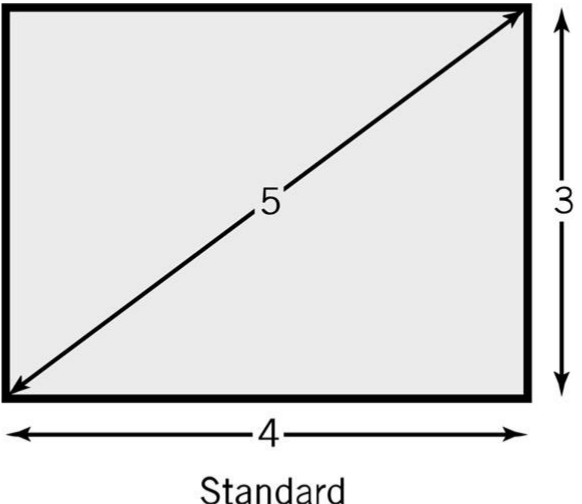


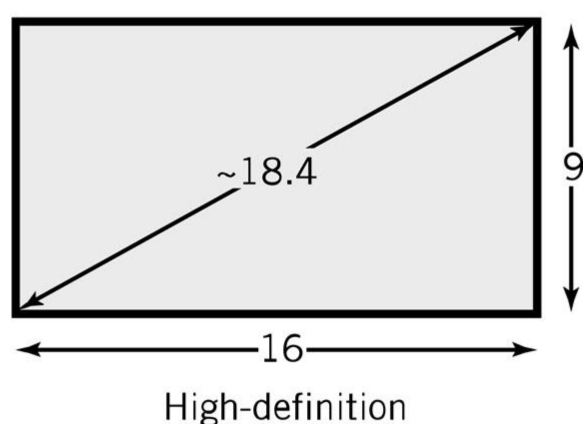
Copyright 2010 John Wiley & Sons, Inc.



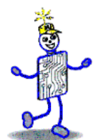
Displays

- Pixel – picture element
- Screen Size: diagonal length of screen
- Aspect ratio – X pixels to Y pixels
  - 4:3 – older displays
  - 16:9 – widescreen displays
- Pixel color is determined by intensity of 3 colors – Red, Green and Blue (RGB)
- True Color – 8 bits for each color
  - 256 levels of intensity for each color
  - 256 * 256 * 256 = 16.7 million colors





Copyright 2010 John Wiley & Sons, Inc.



Resolution and Picture Size

- Resolution
  - Measured as either number of pixels per inch or size of an individual pixel
  - Screen resolution examples:
    - 768 x 1024
    - 1440 x 900
    - 1920 x 1080
- Picture size calculation
  - Resolution * bits required to represent number of colors in picture
  - Example: resolution is 100 pixels by 50 pixels, 4 bits required for a 16 color image

    100 * 50 * 4 bits = 20,000 bits
- Video memory requirements are significant!

Copyright 2010 John Wiley & Sons, Inc.

Interlaced vs. Progressive Scan

Pass

①

②

Interlaced scan

Progressive scan

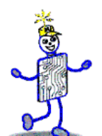
Horizontal retrace

Vertical retrace

Copyright 2010 John Wiley & Sons, Inc.

Diagram of Raster Screen Generation Process

Copyright 2010 John Wiley & Sons, Inc.



Color Transformation Table

123

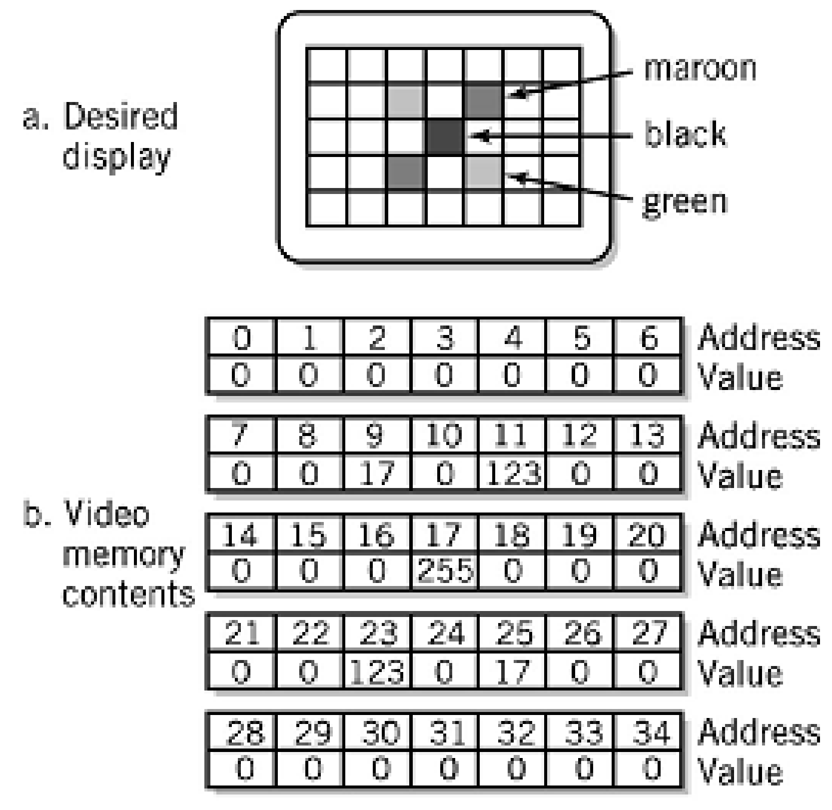
Value of pixel to be displayed

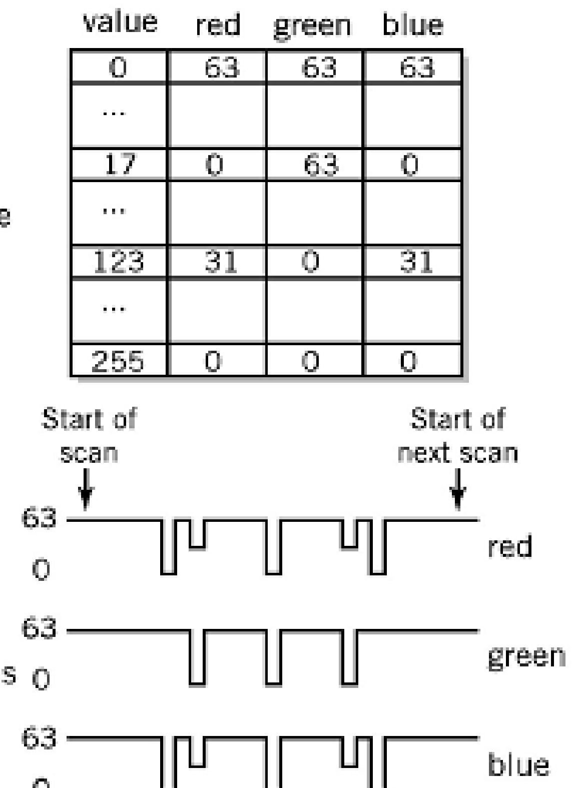
Pixel value	R	G	B
122	32	34	165
123	65	0	177
124	255	0	0

65 to red display | 0 to green display | 177 to blue display

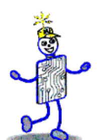
Blue-violet displayed

Display Example





Copyright 2010 John Wiley & Sons, Inc.



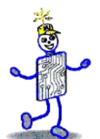
LCD – Liquid Crystal Display

- Fluorescent light or LED panel
- 3 color cells per pixel
- Operation
  - 1st filter polarizes light in a specific direction
  - Electric charge rotates molecules in liquid crystal cells proportional to the strength of colors
  - Color filters only let through red, green, and blue light
  - Final filter lets through the brightness of light proportional to the polarization twist

Copyright 2010 John Wiley & Sons, Inc.

Liquid Crystal Display

Copyright 2010 John Wiley & Sons, Inc.



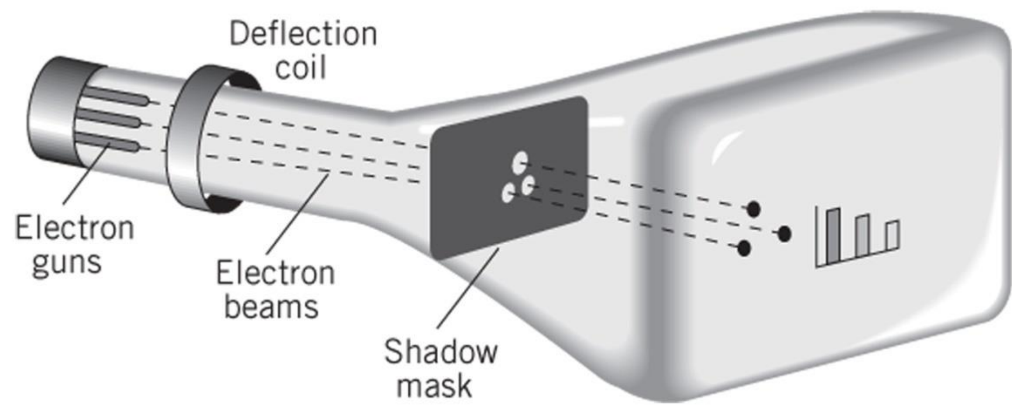
LCDs (continued)

- Active matrix
  - One transistor per cell
  - More expensive
  - Brighter picture
- Passive matrix
  - One transistor per row or column
  - Each cell is lit in succession
  - Display is dimmer since pixels are lit less frequently

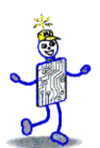
Copyright 2010 John Wiley & Sons, Inc.

CRT Display Technology

- CRTs (similar to TVs)
  - 3 stripes of phosphors for each color
  - 3 separate electron guns for each color
  - Strength of beam → brightness of color
  - Raster scan
    - 30x per second
    - Interlaced vs. non-interlaced (progressive scan)



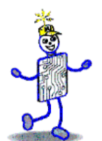
Copyright 2010 John Wiley & Sons, Inc.



OLED Display Technology

- No backlight
- Consists of red, green and blue LEDs
- Each LED lights up individually
- Very thin displays with panels less than 3mm thick!

Copyright 2010 John Wiley & Sons, Inc.



Printers

- Dots vs. pixels
  - 300-2400 dpi vs. 70-100 pixels per inch
  - Dots are on or off, pixels have intensities
- Types
  - Typewriter / Daisy wheels – obsolete
  - Impact printing - dot matrix – mostly obsolete
  - Inkjet – squirts heated droplets of ink
  - Laser printer
  - Thermal wax transfer
  - Dye Sublimation

Copyright 2010 John Wiley & Sons, Inc.

Copyright 2010 John Wiley & Sons, Inc.

Creating a Gray Scale

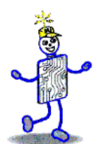
black

dark gray

light gray

white

Copyright 2010 John Wiley & Sons, Inc.

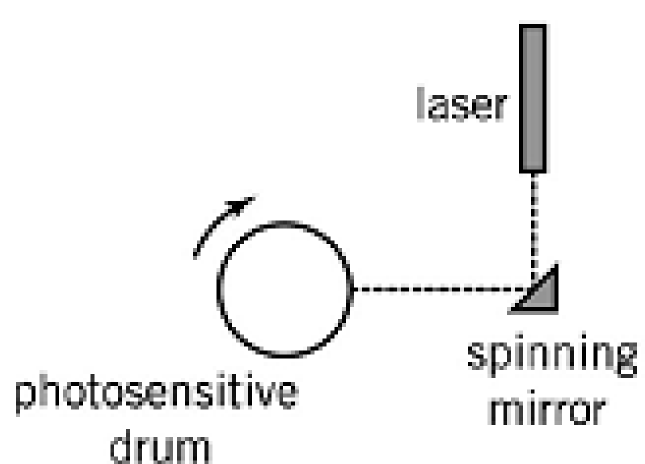


Laser Printer Operation

1. Dots of laser light are beamed onto a drum
2. Drum becomes electrically charged
3. Drum passes through toner which then sticks to the electrically charged places
4. Electrically charged paper is fed toward the drum
5. Toner is transferred from the drum to the paper
6. The fusing system heats and melts the toner onto the paper
7. A corona wire resets the electrical charge on the drum

Copyright 2010 John Wiley & Sons, Inc.

Laser Printer Operation



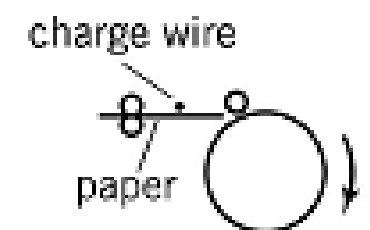
1. A laser is fired in correspondence to the dots that are to be printed. A spinning mirror causes the dots to be fanned out across the drum. The drum rotates to create the next line, usually 1/300th or 1/600th of an inch.

   The drum is photosensitive. As a result of the laser light, the drum will become electrically charged wherever a dot is to be printed.

2. As the drum continues to rotate, the charged part of the drum passes through a tank of black powder called toner. Toner sticks to the drum wherever the charge is present. Thus, it looks like the image.

Copyright 2010 John Wiley & Sons, Inc.

Laser Printer Operation

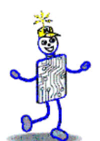


3. A sheet of paper is fed toward the drum. A charge wire coats the paper with electrical charges. When it contacts the drum, it picks up the toner from the drum.

4. As the paper rolls from the drum, it passes over a heat and pressure area known as the fusing system. The fusing system melts the toner to the paper. The printed page then exits the printer.

   At the same time, the surface of the drum passes over another wire, called a corona wire. This wire resets the charge on the drum, to ready it for the next page.

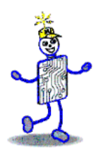
Copyright 2010 John Wiley & Sons, Inc.



Other Computer Peripherals

- Scanners
  - Flatbed, sheet-fed, hand-held
  - Light is reflected off the sheet of paper
- User Input Devices
  - Keyboard, mouse, light pens, graphics tablets
- Communication Devices
  - Telephone modems
  - Network devices

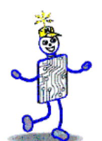
Copyright 2010 John Wiley & Sons, Inc.



Network Communication Devices

- Network is just another I/O device
- Network I/O controller is the network interface card (NIC)
- Types of network connections
  - Ethernet, FDDI fiber, token-ring
- Medium access control (MAC) protocols
  - Define the specific rules of communication for the network

Copyright 2010 John Wiley & Sons, Inc.



Copyright 2010 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein."