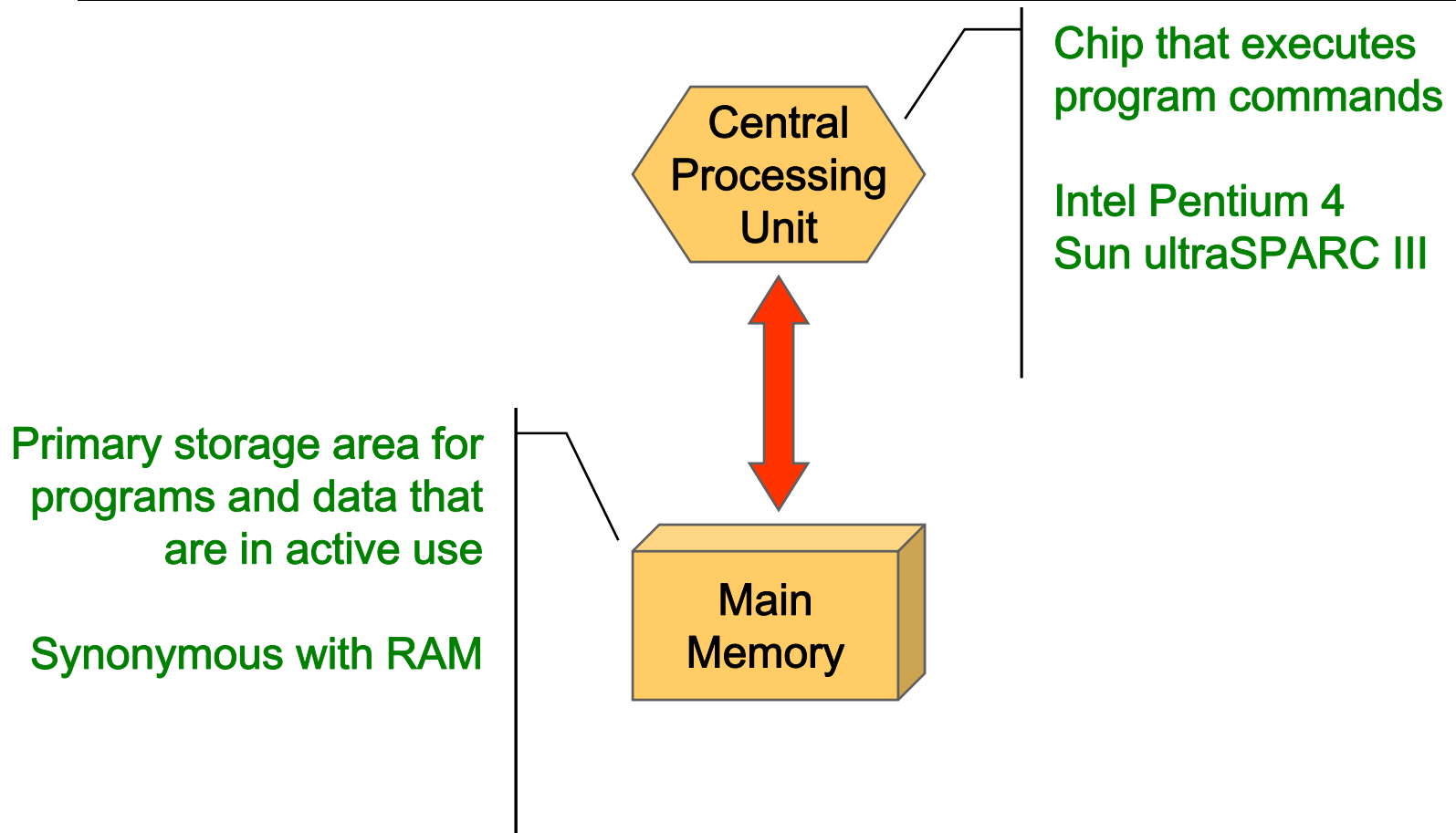




Introduction to Computer System and Architecture

CPU and Main Memory

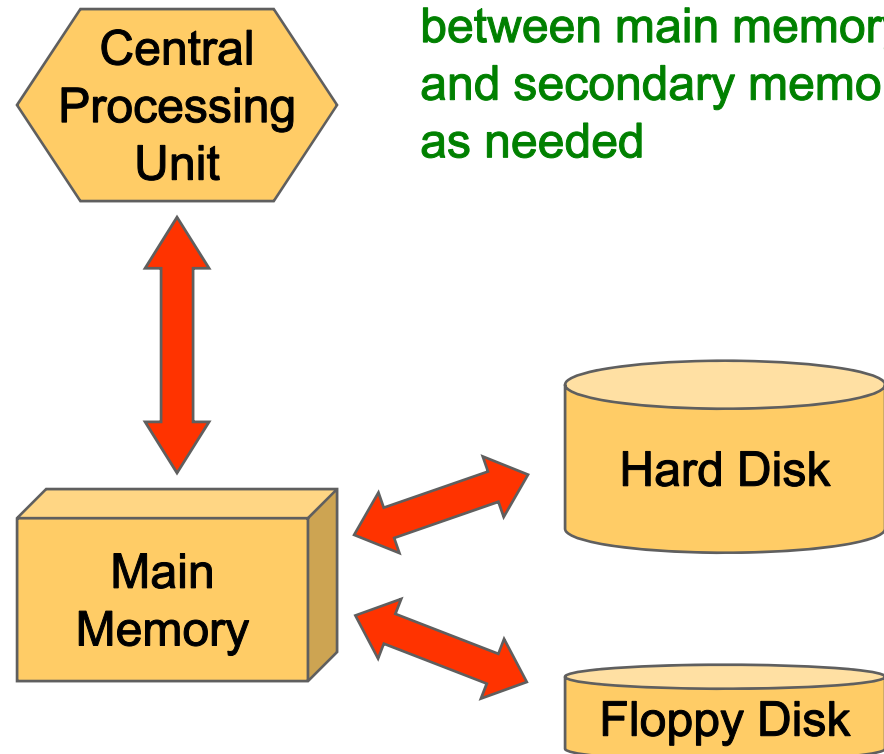


Secondary Memory Devices

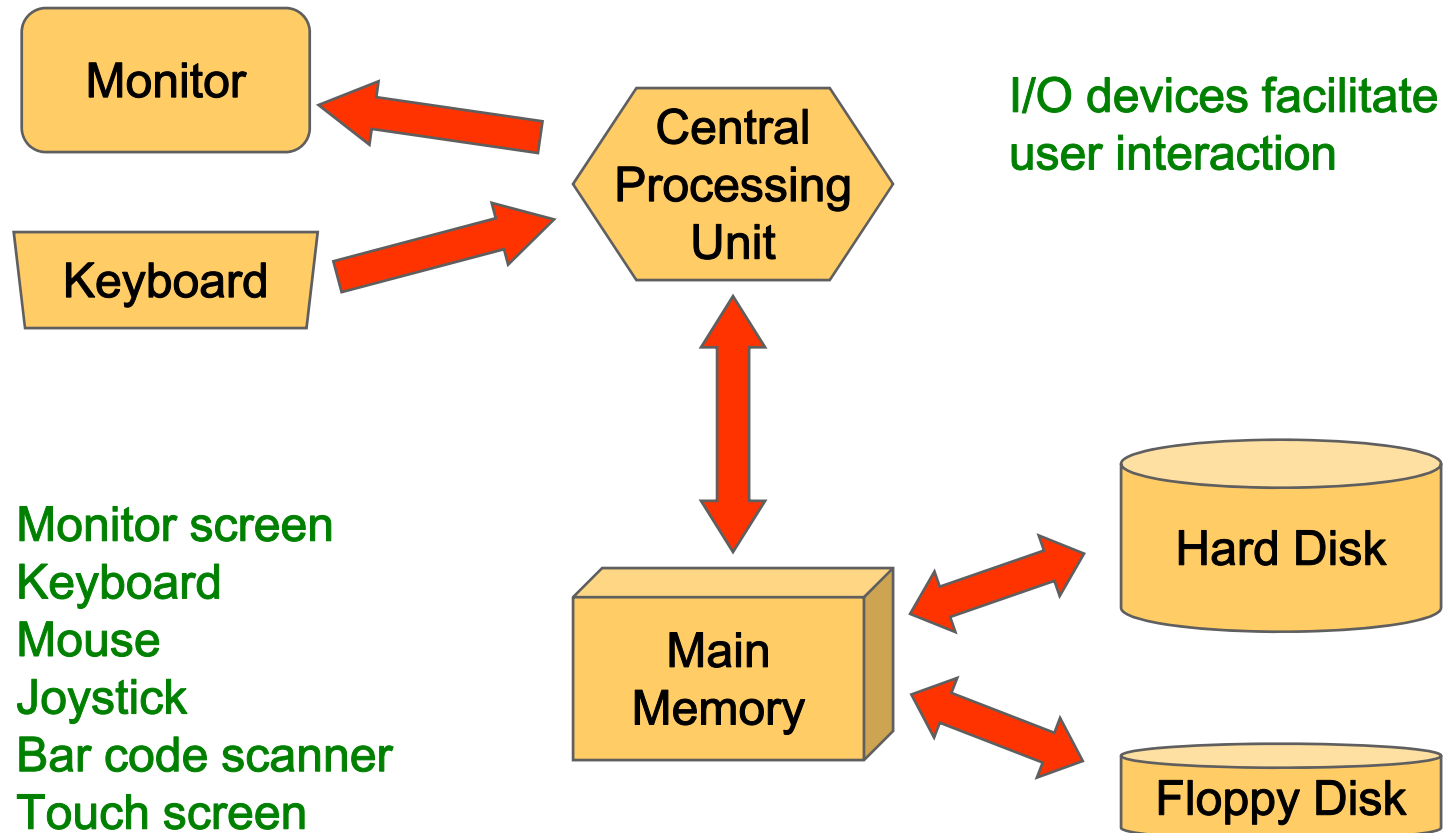
Secondary memory devices provide long-term storage

Information is moved between main memory and secondary memory as needed

Hard disks
Floppy disks
ZIP disks
Writable CDs
Writable DVDs
Tapes



Input / Output Devices



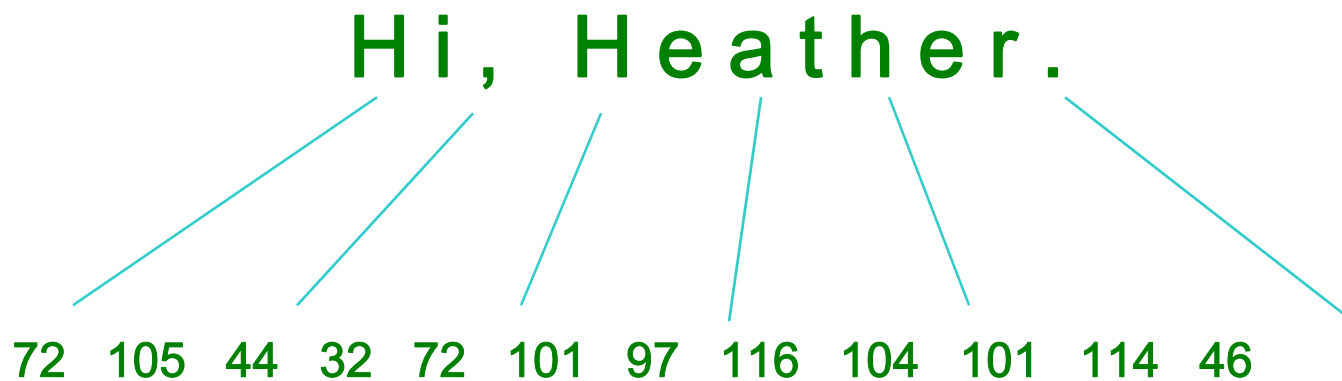


Software Categories

- Operating System
 - controls all machine activities
 - provides the user interface to the computer
 - manages resources such as the CPU and memory
 - Windows XP, Unix, Linux, Mac OS
- Application program
 - generic term for any other kind of software
 - word processors, missile control systems, games
- Most operating systems and application programs have a *graphical user interface* (GUI)

Representing Text Digitally

- Every character is stored as a number, including spaces, digits, and punctuation
- Corresponding upper and lower case letters are separate characters



- A common code is ASCII (American Standard Code for Information Interchange).



Binary Numbers

- Once information is digitized, it is represented and stored in memory using the *binary number system*
- A single binary digit (0 or 1) is called a *bit*
- Devices that store and move information are cheaper and more reliable if they have to represent only two states
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)
- Permutations of bits are used to store values

Bit Permutations

- Each permutation can represent a particular item
- There are 2^N permutations of N bits
- Therefore, N bits are needed to represent 2^N unique items

**How many
items can be
represented by**

1 bit ?	$2^1 = 2$ items
2 bits ?	$2^2 = 4$ items
3 bits ?	$2^3 = 8$ items
4 bits ?	$2^4 = 16$ items
5 bits ?	$2^5 = 32$ items

Storing Information

9278

9279

9280

9281

9282

9283

9284

9285

9286



A vertical stack of eight rectangular boxes representing memory cells. The boxes are arranged in a column. The second box from the top contains the binary value '10011010'. The third and fourth boxes are empty. The fifth and sixth boxes are filled with a solid cyan color. The seventh and eighth boxes are empty. Above the stack are three vertically aligned dots, and below the stack are three vertically aligned dots.

10011010

Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

Large values are stored in consecutive memory locations



Storage Capacity

- Every memory device has a *storage capacity*, indicating the number of bytes it can hold
- Capacities are expressed in various units:

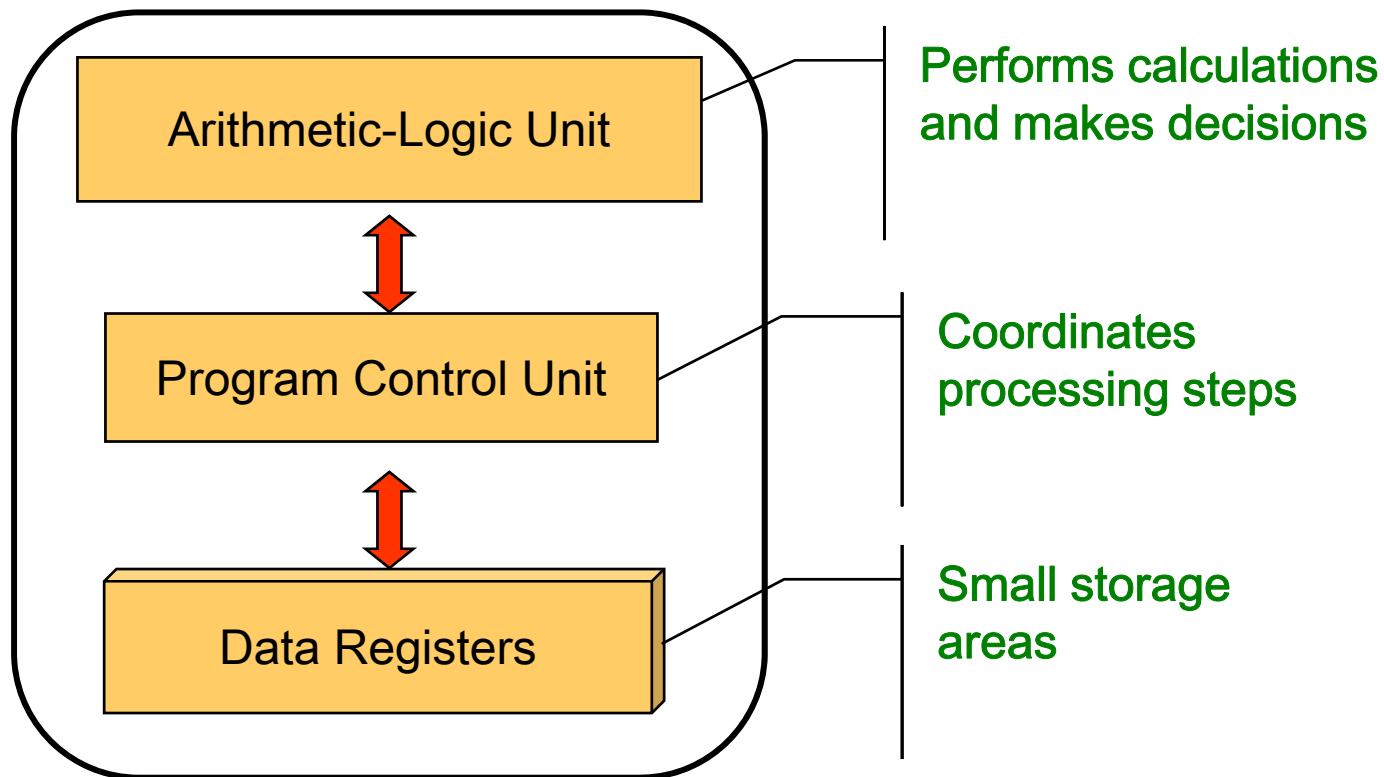
<u>Unit</u>	<u>Symbol</u>	<u>Number of Bytes</u>
kilobyte	KB	$2^{10} = 1024$
megabyte	MB	2^{20} (over 1 million)
gigabyte	GB	2^{30} (over 1 billion)
terabyte	TB	2^{40} (over 1 trillion)



Memory

- Main memory is *volatile* - stored information is lost if the electric power is removed
- Secondary memory devices are *nonvolatile*
- Main memory and disks are *direct access* devices - information can be reached directly
- The terms *direct access* and *random access* often are used interchangeably
- A magnetic tape is a *sequential access* device since its data is arranged in a linear order - you must get by the intervening data in order to access other information

The Central Processing Unit (CPU)



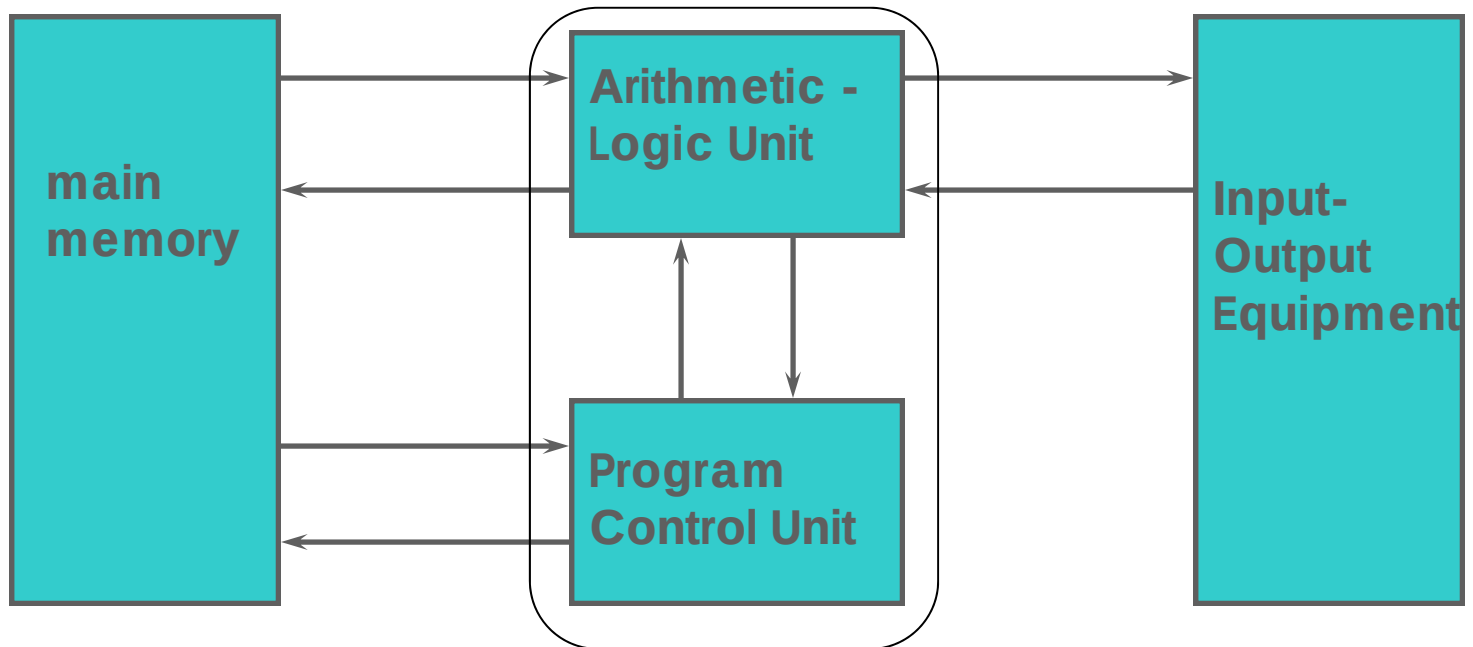
CPU - Microprocessor

- A CPU is on a chip called a *microprocessor*



Simple Machine Organization

- von Neumann machine
 - ALU performs transfers between memory and I/O devices



Simple CPU (no data registers)



CPU

- A CPU is a sequential circuit
- It repeatedly reads and executes an **instruction** from its memory
- Machine language program is a set of instructions drawn from CPU instruction set.
- An instruction is composed of one instruction **operation**, and a number of **operands**
 - Operand means the data needed for the operation
 - For example:
 - 1-operand format: use the accumulator (AC) register
 - e.g. Load A (It means $AC \leftarrow A$)



CPU speed

- The speed of a CPU is controlled by the *system clock*
- The system clock generates an electronic pulse at regular intervals
- The pulses coordinate the activities of the CPU
- The speed is usually measured in *gigahertz* (GHz)

e.g. 2.8 GHz means 2.8 billion pulses per second

Von Neumann Architecture

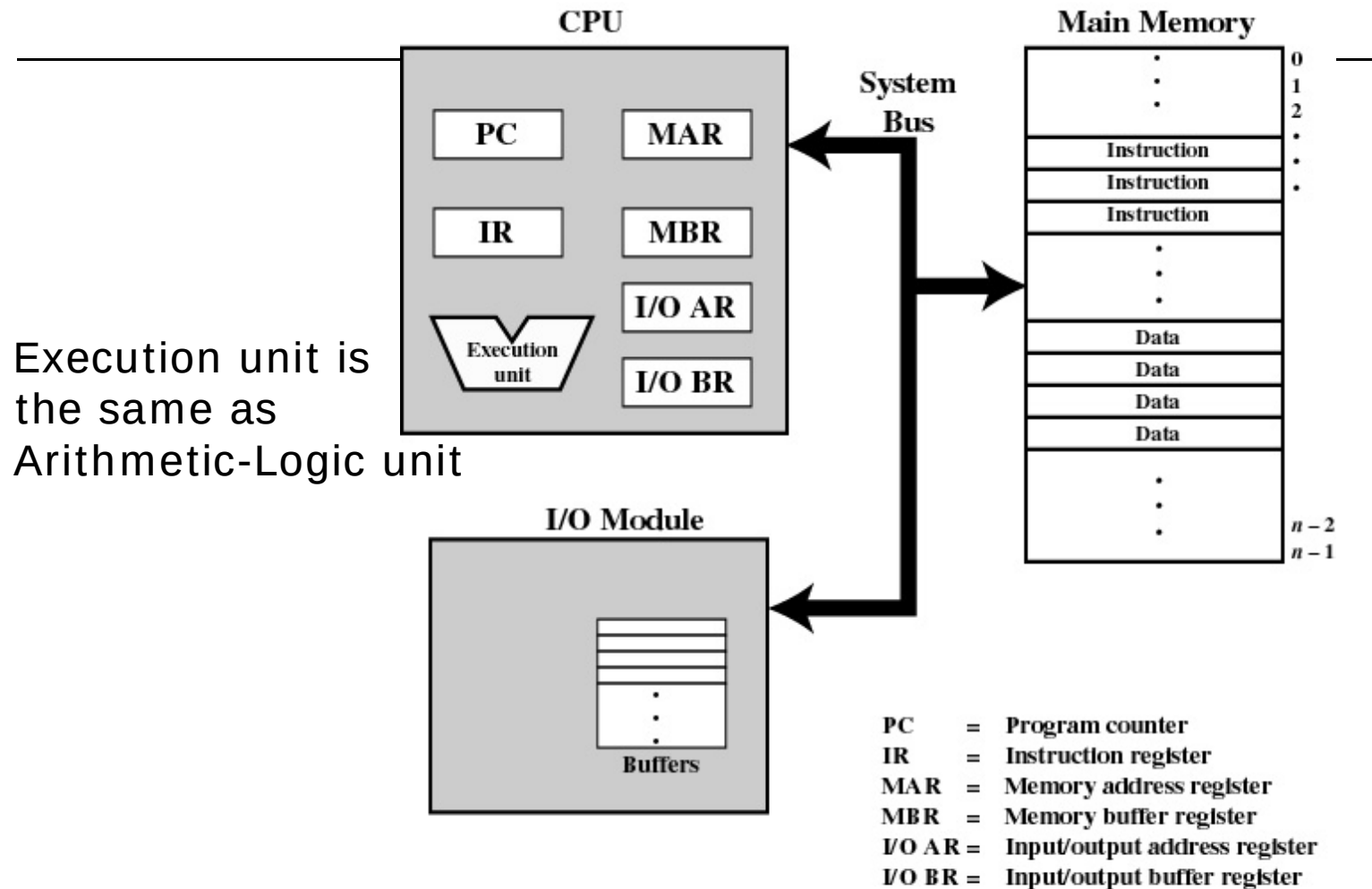


Figure 1.1 Computer Components: Top-Level View
SEEM 3460



Type of Instructions

- Processor-memory
 - transfer data between processor (register) and memory
- Processor-I/O
 - data transferred to or from a peripheral device
- Data processing
 - arithmetic or logic operation on data
- Control
 - alter sequence of execution

Instruction Set Principles

- Multiple data registers in CPU
- Data registers can be used to hold intermediate results during execution of instructions
- They are instruction-visible
e.g. R1, R2, R3
- 2-operand format: one of the operands is both a source and result
e.g. Add R1,B (It means $R1 \leftarrow R1+B$)
- 3-operand format: result and two source operands
e.g. Add R3,R1,R2 (It means $R3 \leftarrow R1+R2$)

Different Kinds of Instruction Sets

- Different kinds of computer architecture and the corresponding instruction sets

	1-operand format	2-operand format	3-operand format
	(AC) Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store C,R1	Add R3,R1,R2
Pop C			Store C,R3

FIGURE 2.1 The code sequence for $C = A + B$ for four instruction sets.

- Examples of instruction operation: Load, Add, Store
- Examples of memory address: A, B, C
- Examples of registers: Accumulator (AC), R1, R2, R3

Example of Program Compilation

- Suppose there is a C statement in the source program:

$A = A + B$

- Suppose the CPU supports single-operand accumulator-kind of instructions.
- After compilation, this statement is translated into 3 instructions:

LOAD B	($AC \leftarrow \text{memory address B}$)
ADD A	($AC \leftarrow AC + \text{memory address A}$)
STORE A	($\text{memory address A} \leftarrow AC$)

- LOAD B means that it loads the content referenced by address B to the internal accumulator register AC
- ADD A means that it adds the content referenced by address A to the internal accumulator register AC
- STORE A means that it stores the content of the internal accumulator register AC into the memory location referenced by address A



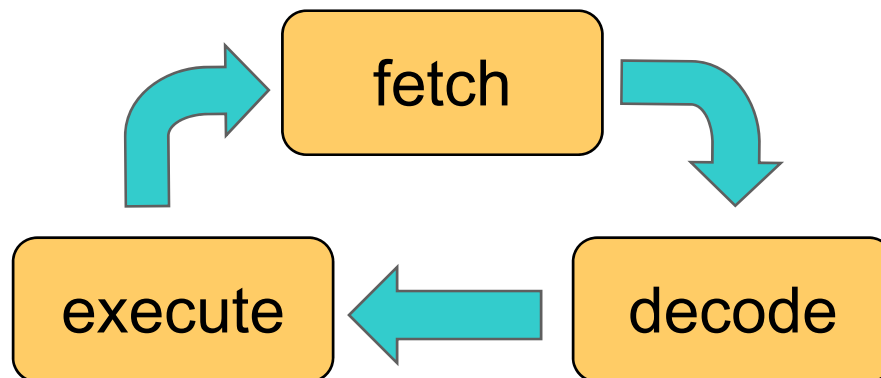
Instruction Set Principles – Control Registers

- Used by processor to control the operation of the processor
- Program Counter (PC)
 - Contains the address of an instruction to be fetched
- Instruction Register (IR)
 - Contains the instruction most recently fetched

Instruction Cycle

- The CPU continuously follows the *fetch-decode-execute cycle*:

Retrieve an instruction from main memory



Carry out the instruction

Determine what the instruction is



Instruction Fetch and Execute

- Program counter (PC) holds the address of the instruction to be fetched next.
- The processor fetches the instruction from memory.
- Program counter is incremented by one unit after each fetch so that the processor knows where the next instruction is.
- Fetched instruction is placed in the instruction register (IR).



Example of Program Execution

- Recall that the C statement $A = A + B$ has been compiled into the following instructions:
 - LOAD B ($AC \leftarrow \text{memory address B}$)
 - ADD A ($AC \leftarrow AC + \text{memory address A}$)
 - STORE A ($\text{memory address A} \leftarrow AC$)
- When the user executes the executable file, the operating system will:
 - place those three instructions starting from a memory address such as 300
 - allocate a memory address such as 940 for the variable B
 - allocate a memory address such as 941 for the variable A

Example of Program Execution

0001 1001 0100 0000
1 9 4 0
LOAD 940

0101 1001 0100 0001
5 9 4 1
ADD 941

0010 1001 0100 0001
2 9 4 1
STORE 941

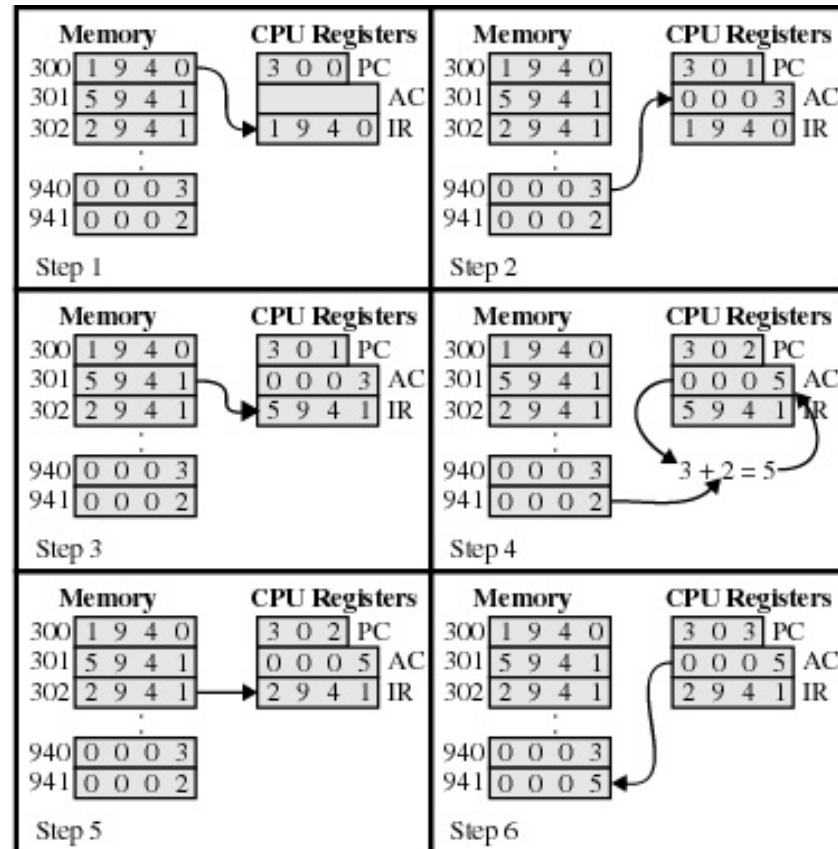


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

Instruction Set Principles – Instruction Encoding Examples

Operation & no. of operands	Address specifier 1	Address field 1	...	Address specifier n	Address field n
--------------------------------	------------------------	--------------------	-----	------------------------	--------------------

(a) Variable (e.g., VAX)

Operation	Address field 1	Address field 2	Address field 3
-----------	--------------------	--------------------	--------------------

(b) Fixed (e.g., DLX, MIPS, Power PC, Precision Architecture, SPARC)

Operation	Address specifier	Address field
-----------	----------------------	------------------

Operation	Address specifier 1	Address specifier 2	Address field
-----------	------------------------	------------------------	------------------

Operation	Address specifier	Address field 1	Address field 2
-----------	----------------------	--------------------	--------------------

(c) Hybrid (e.g., IBM 360/70, Intel 80x86)



Instruction Set Principles – Status Registers

- Used for controlling the execution of instructions
- Program Status Word (PSW)
 - Condition Codes or Flags
 - Bits set by the processor hardware as a result of operations
 - Can be accessed by a program but not altered
 - e.g. positive result, negative result, zero, overflow

Examples of Control Flow Instructions

Example instruction	Instruction name	Meaning
J name	Jump	$PC \leftarrow \text{name}; ((PC+4) - 2^{25}) \leq \text{name} < ((PC+4) + 2^{25})$
JAL name	Jump and link	$R31 \leftarrow PC+4; PC \leftarrow \text{name};$ $((PC+4) - 2^{25}) \leq \text{name} < ((PC+4) + 2^{25})$
JALR R2	Jump and link register	$\text{Regs}[R31] \leftarrow PC+4; PC \leftarrow \text{Regs}[R2]$
JR R3	Jump register	$PC \leftarrow \text{Regs}[R3]$
BEQZ R4, name	Branch equal zero	if $(\text{Regs}[R4] == 0)$ $PC \leftarrow \text{name};$ $((PC+4) - 2^{15}) \leq \text{name} < ((PC+4) + 2^{15})$
BNEZ R4, name	Branch not equal zero	if $(\text{Regs}[R4] \neq 0)$ $PC \leftarrow \text{name};$ $((PC+4) - 2^{15}) \leq \text{name} < ((PC+4) + 2^{15})$

FIGURE 2.24 Typical control-flow instructions in DLX.