Computer Fundamentals

Computer

Why we use Computer?

How Computer Works?

Model of Computer

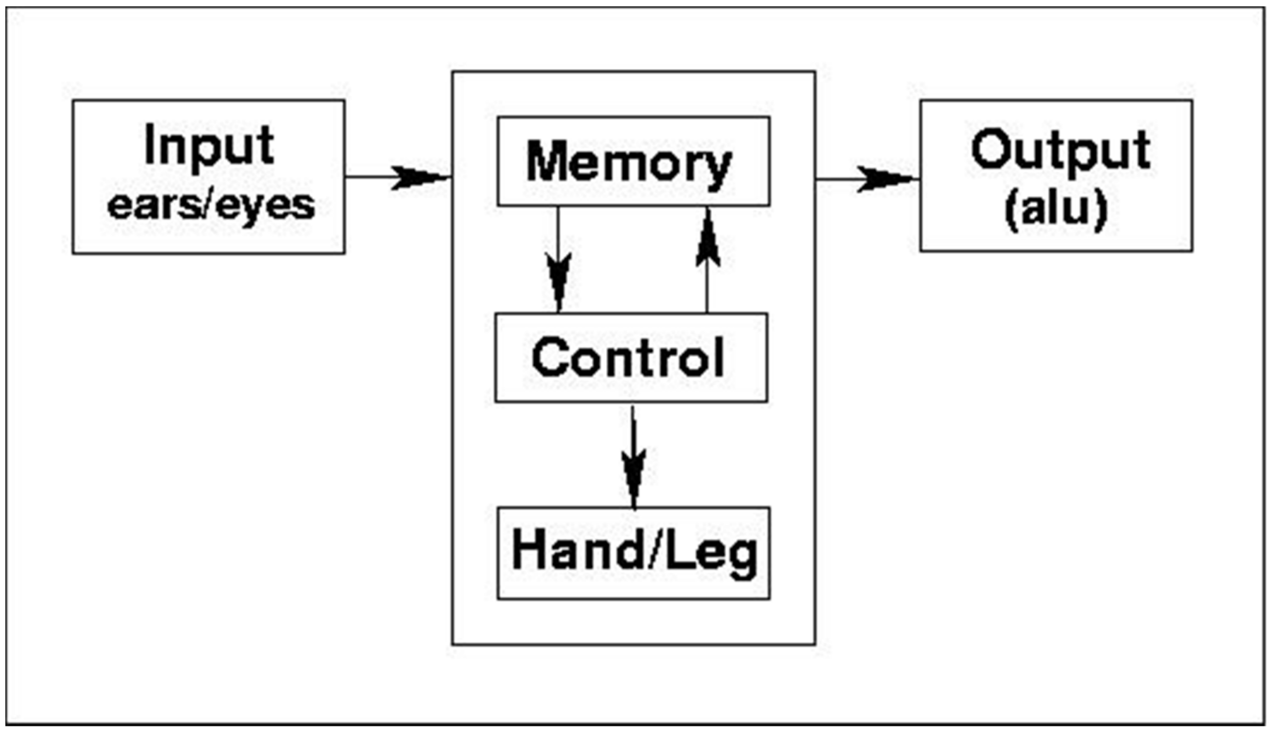
How Computer Works?

Work: Buy 1kg Alu

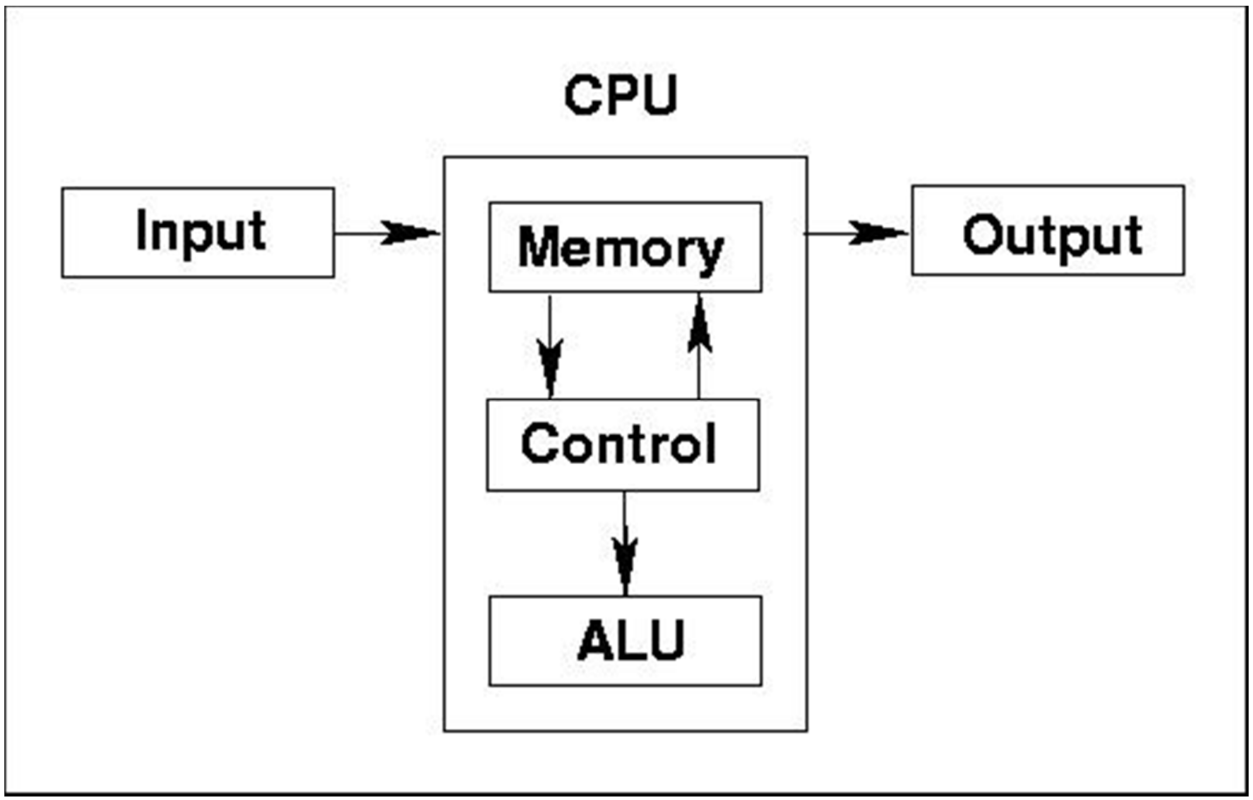
Action:

1. Take Bag, Money
2. Goto Market
3. Search for good Alu
4. Buy 1kg Alu
5. Go home

Model



Computer Model

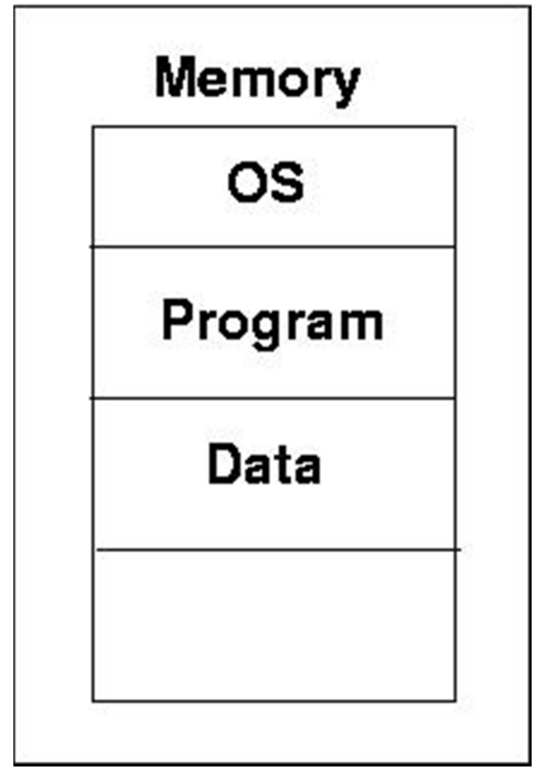


Algorithm: Procedure/Method to achieve desired result

Computer Program:

Set of Instructions

Executes in Sequence



Body --- Hardware

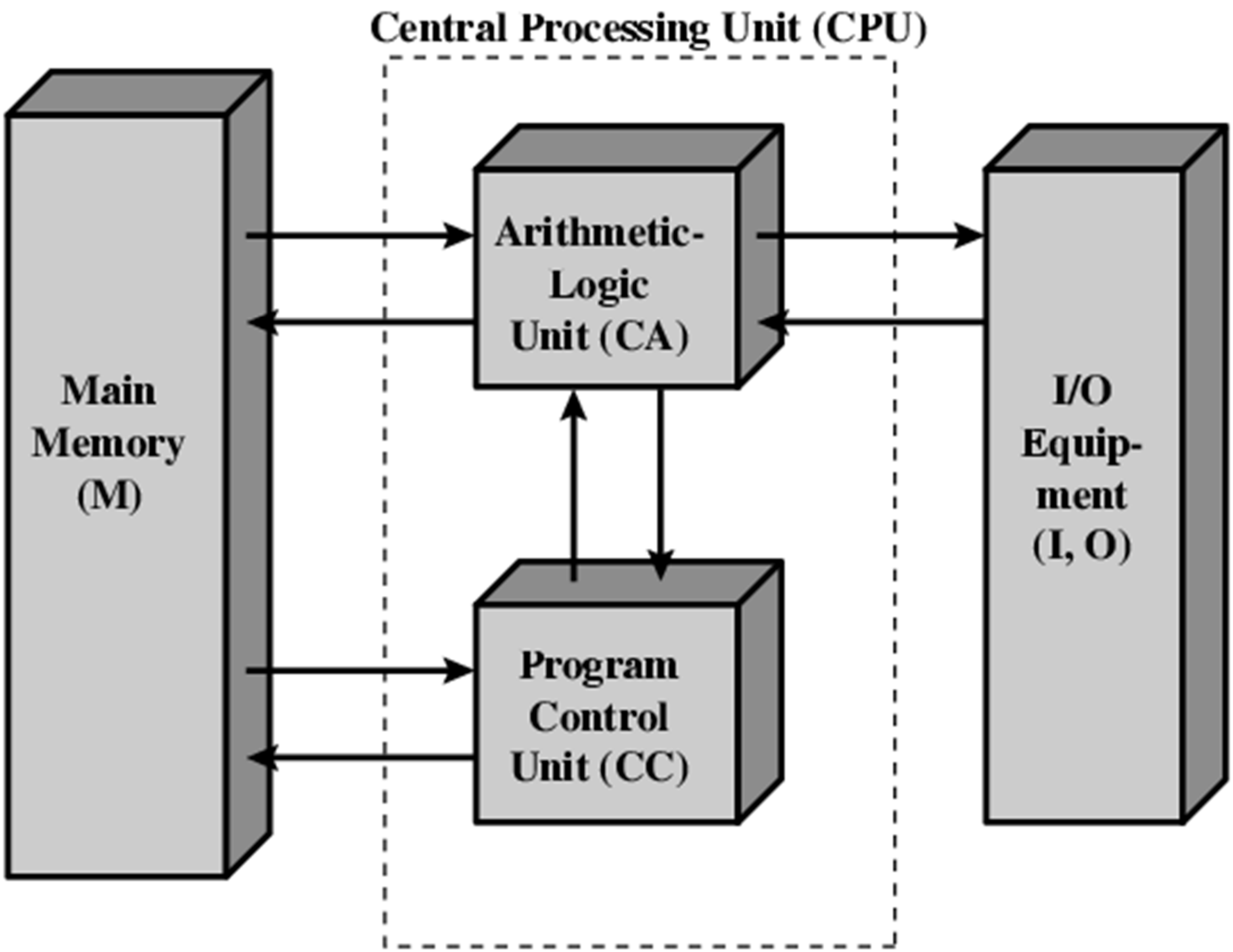
Life --- Software

- **Operating System, Compiler, editor, other tools,** Application Software

von Neumann

- Stored Program concept
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
  - IAS
- Completed 1952

Structure of von Neumann machine



Architecture & Organization 1

- Architecture is those attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - e.g. Is there a multiply instruction?
- Organization is how features are implemented
  - Control signals, interfaces, memory technology.
  - e.g. Is there a hardware multiply unit or is it done by repeated addition?

Architecture & Organization 2

- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture

- This gives code compatibility
  - At least backwards
- Organization differs between different versions

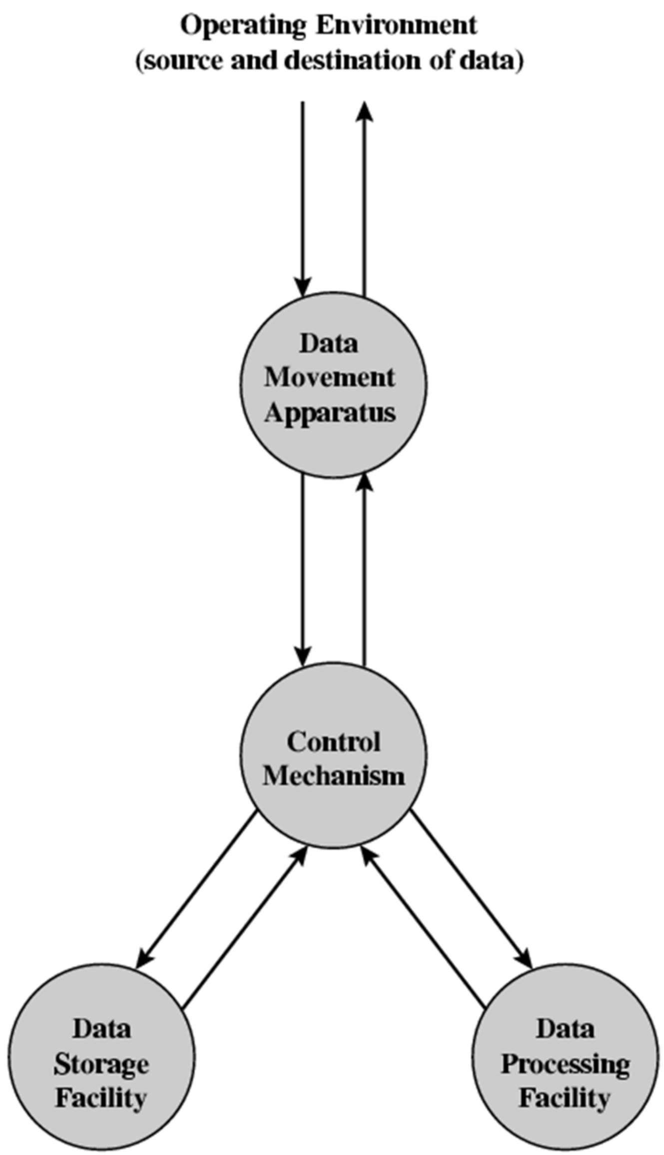
Computer : Structure & Function

- Structure is the way in which components relate to each other
- Function is the operation of individual components as part of the structure

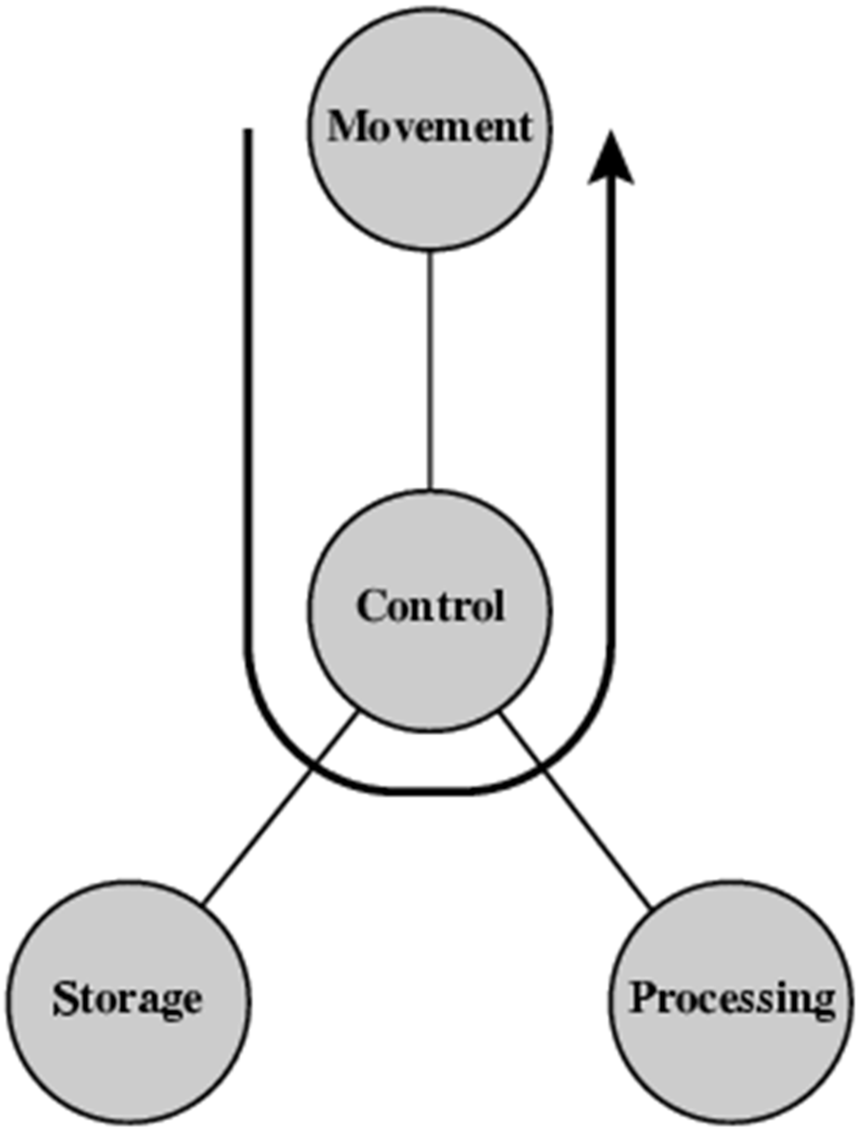
Function

- All computer functions are:
  - Data processing
  - Data storage
  - Data movement
  - Control

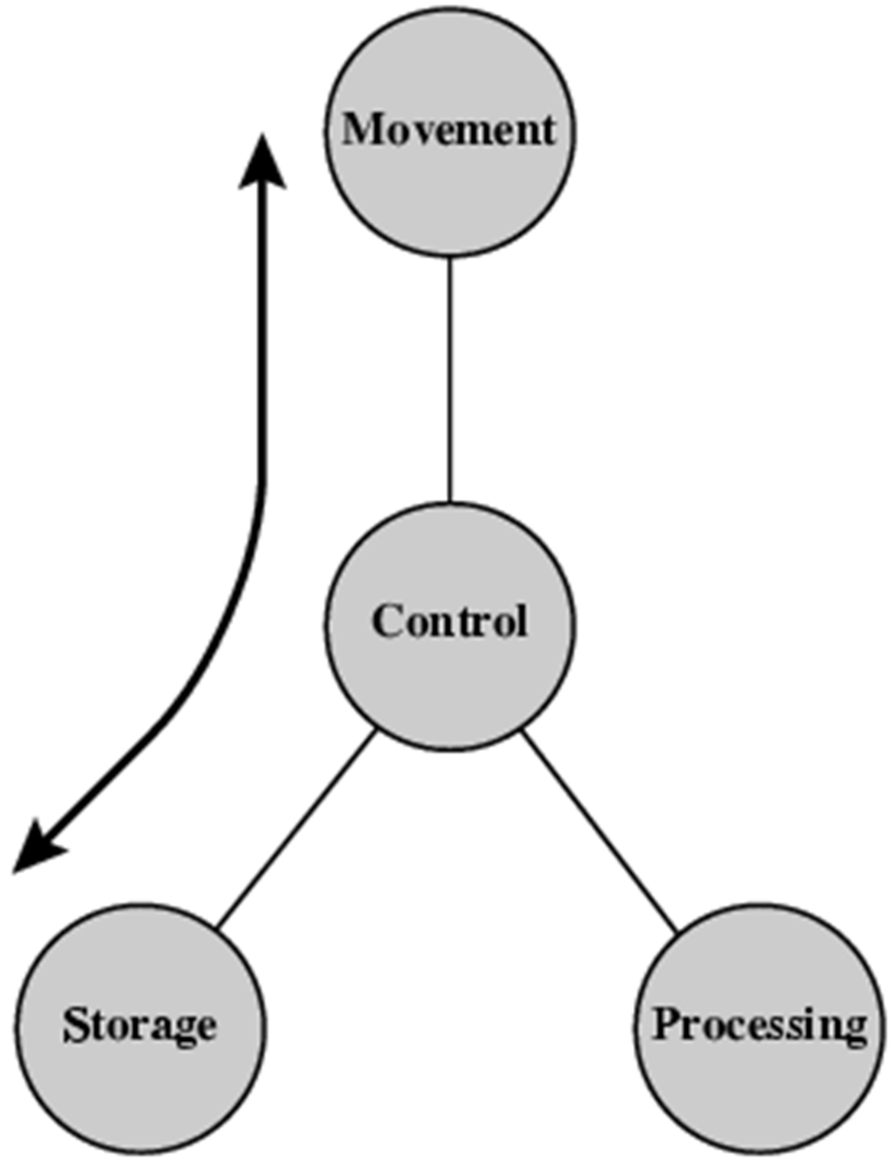
Functional View



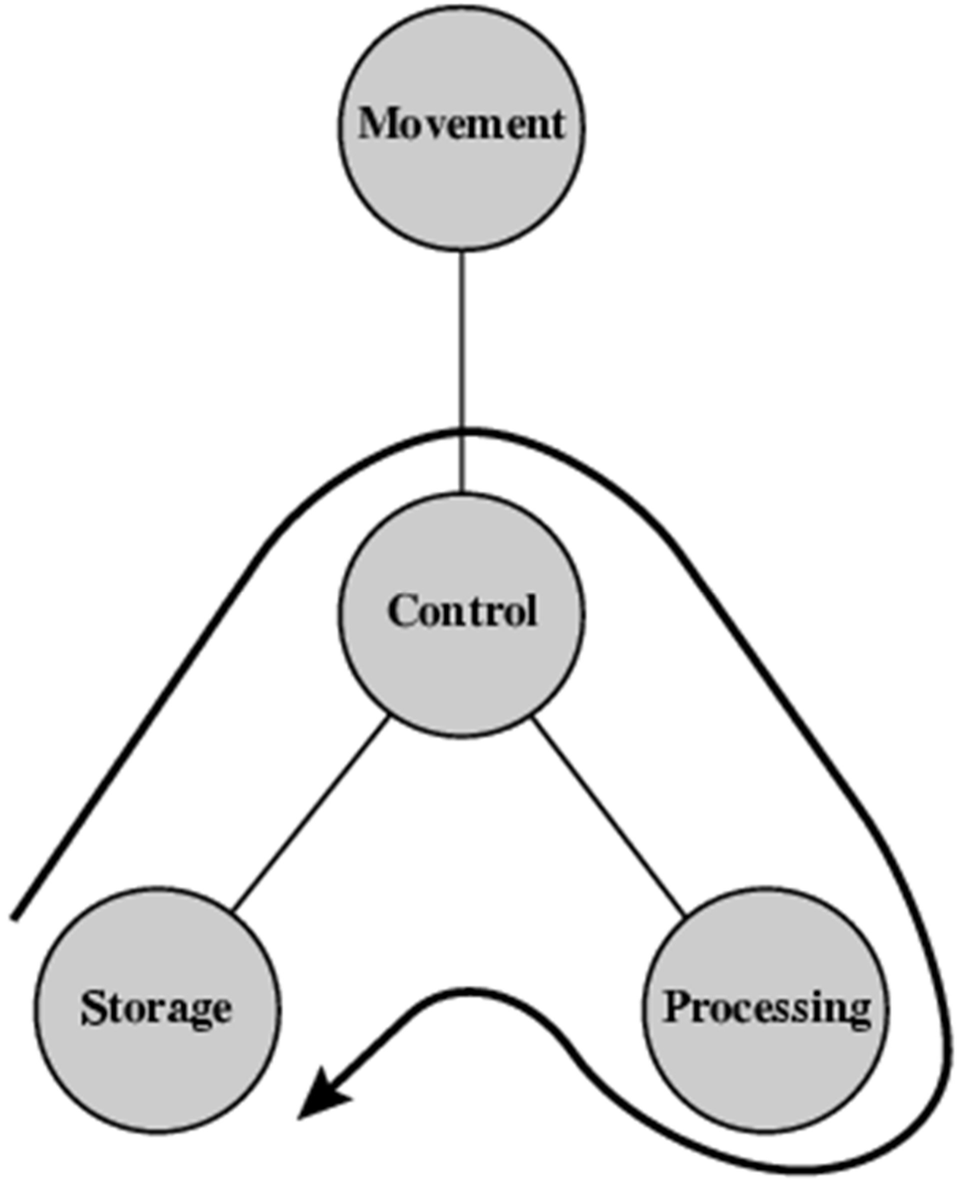
Operations (a) Data movement



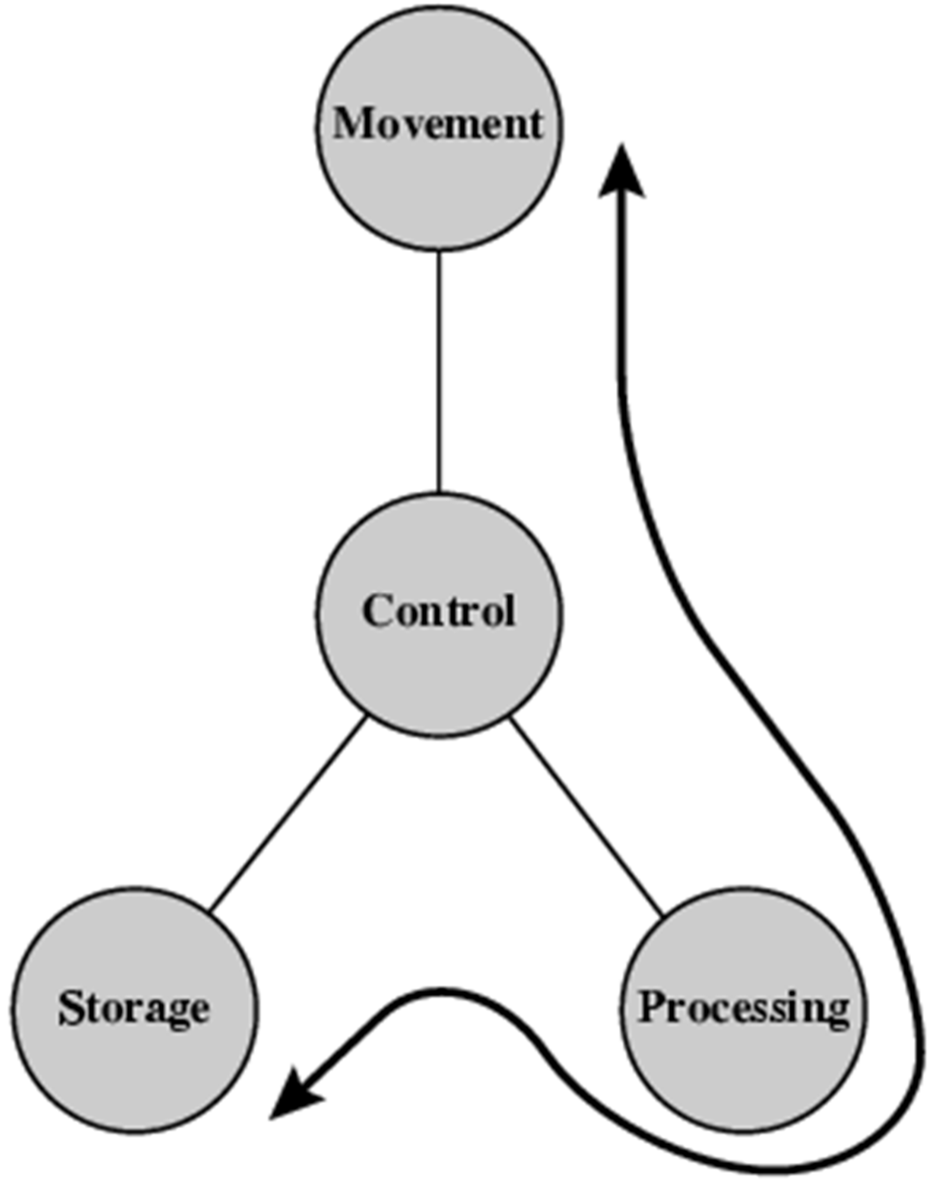
Operations (b) Storage



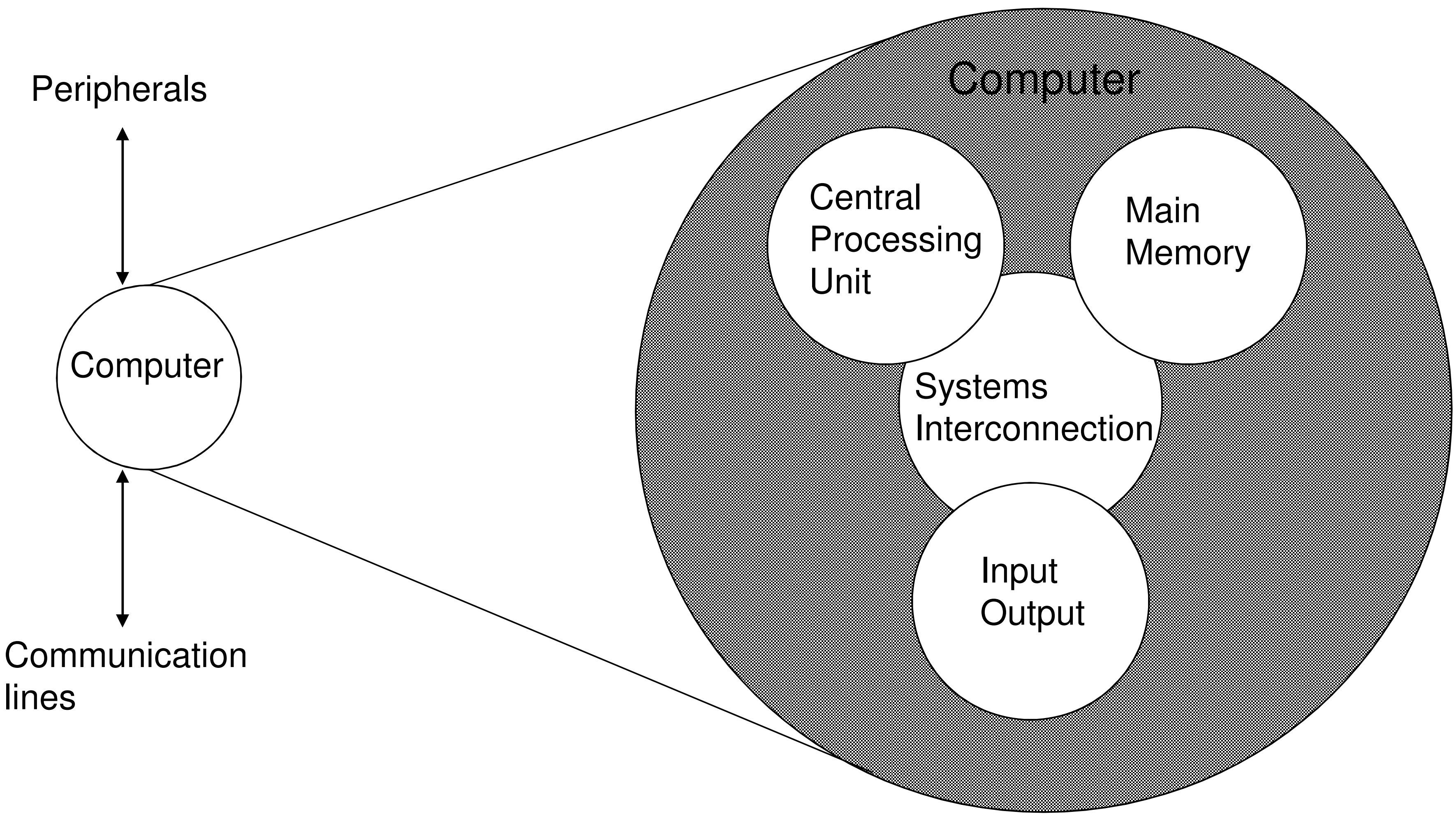
Operation (c) Processing from/to storage



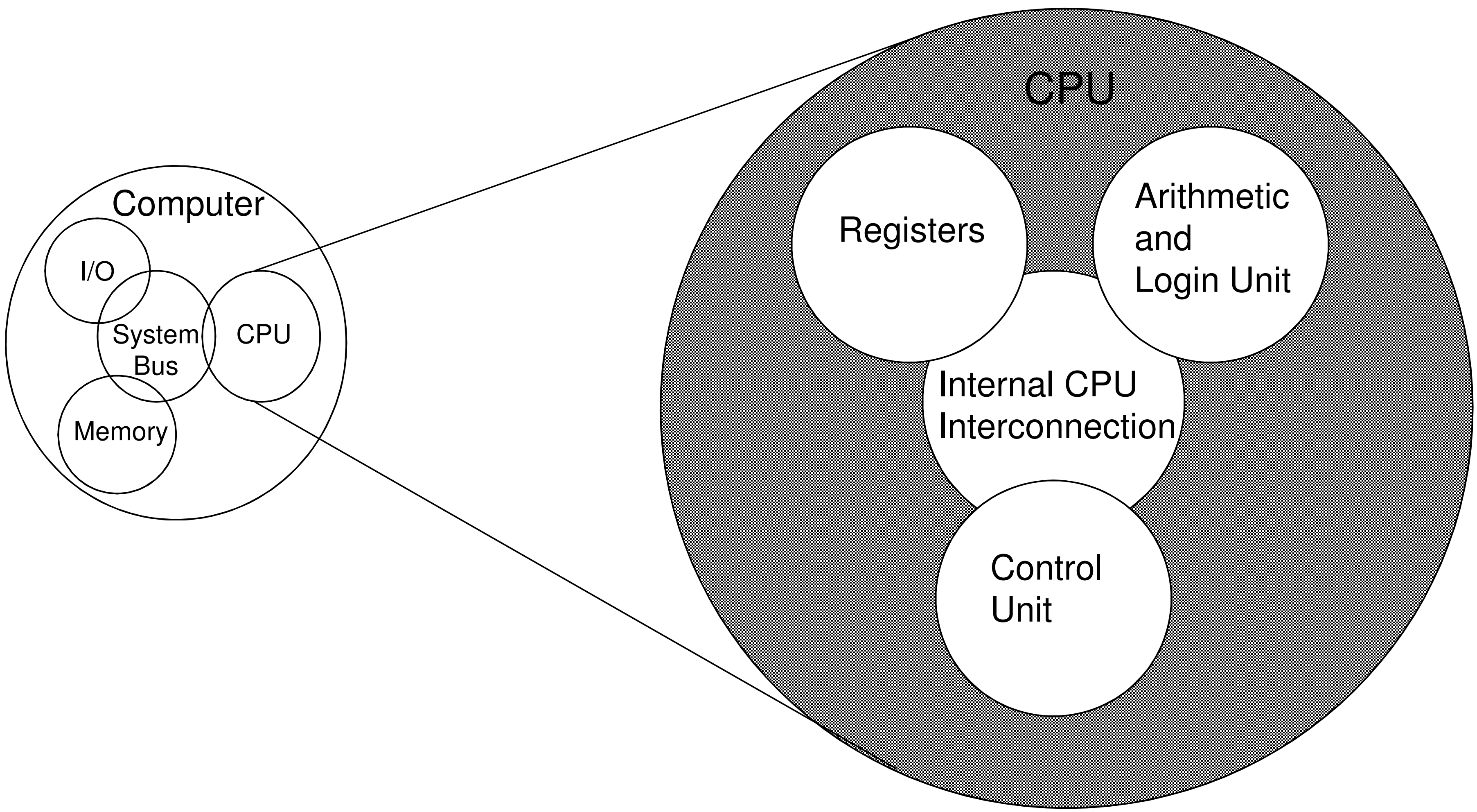
Operation (d)
Processing from storage to I/O



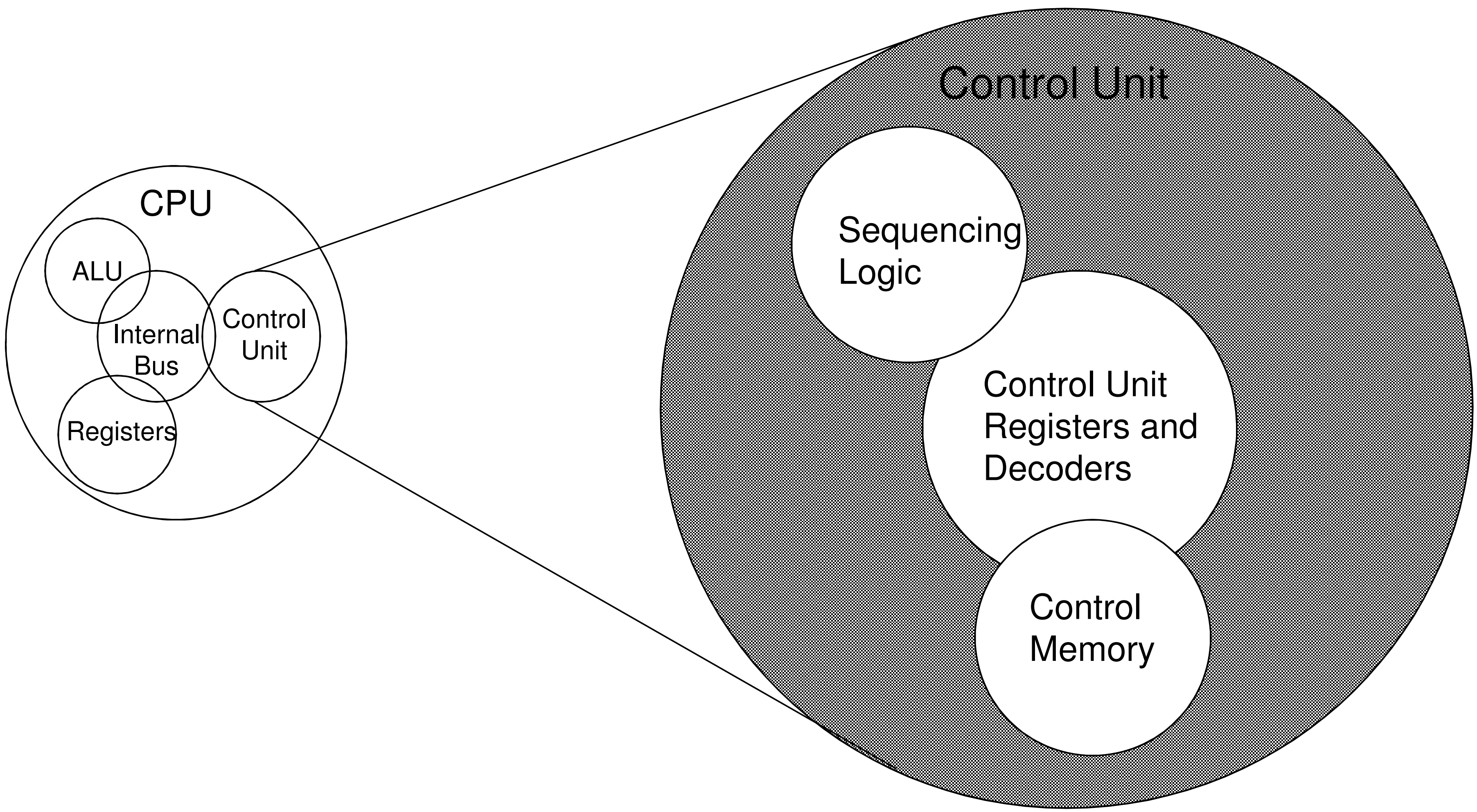
Structure - Top Level



Structure - The CPU



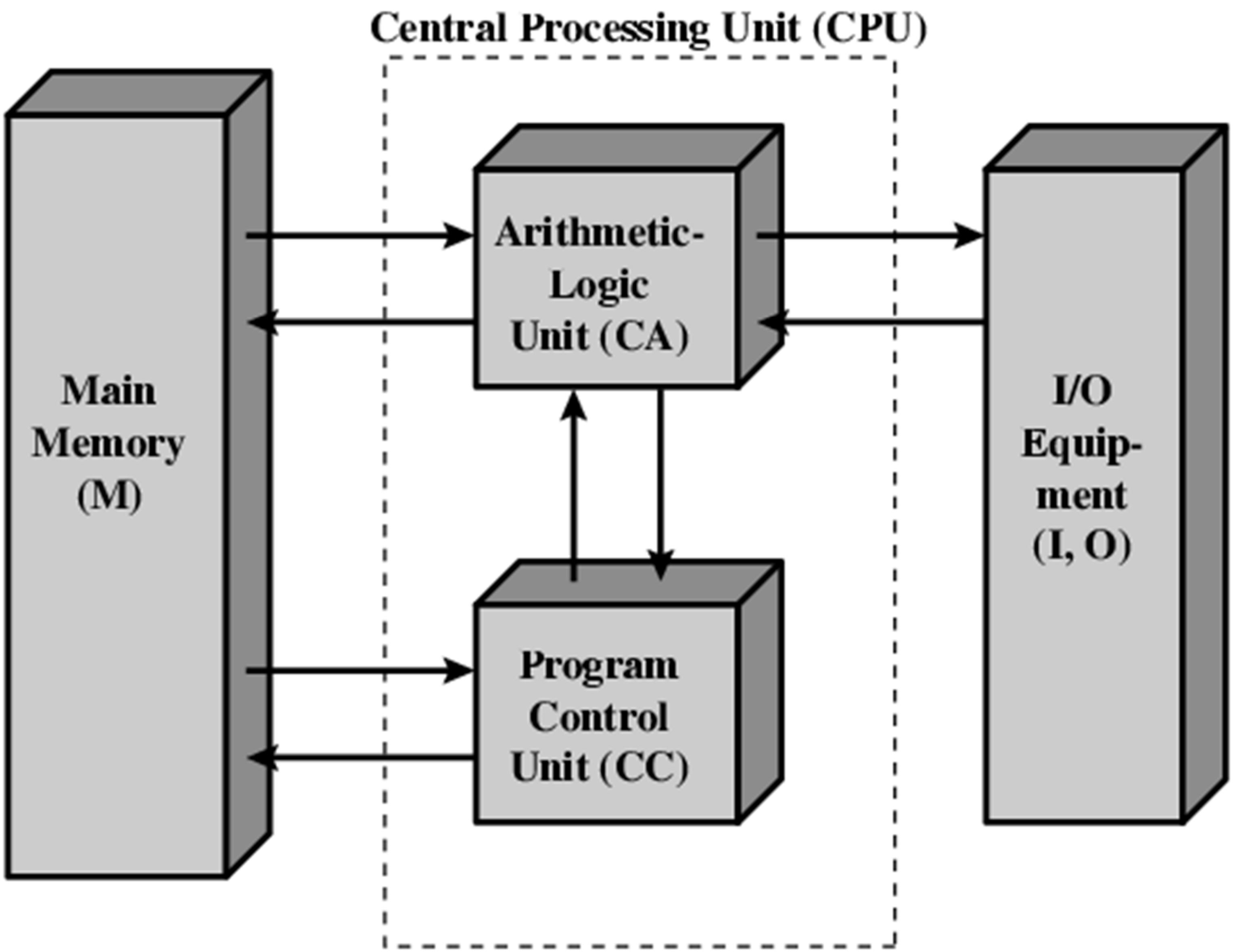
Structure - The Control Unit



von Neumann

- Stored Program concept
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
    - IAS
- Completed 1952

Structure of von Neumann machine

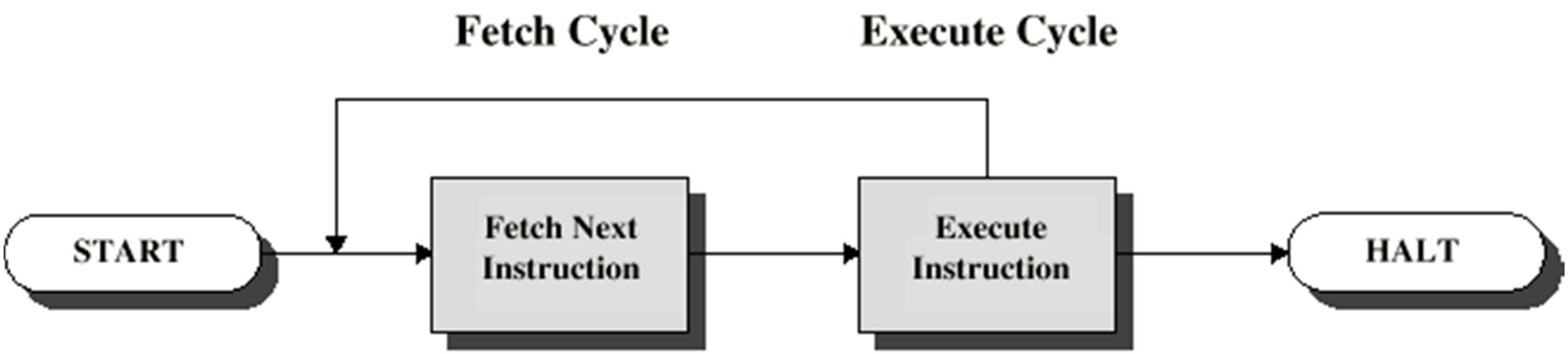


What is a program?

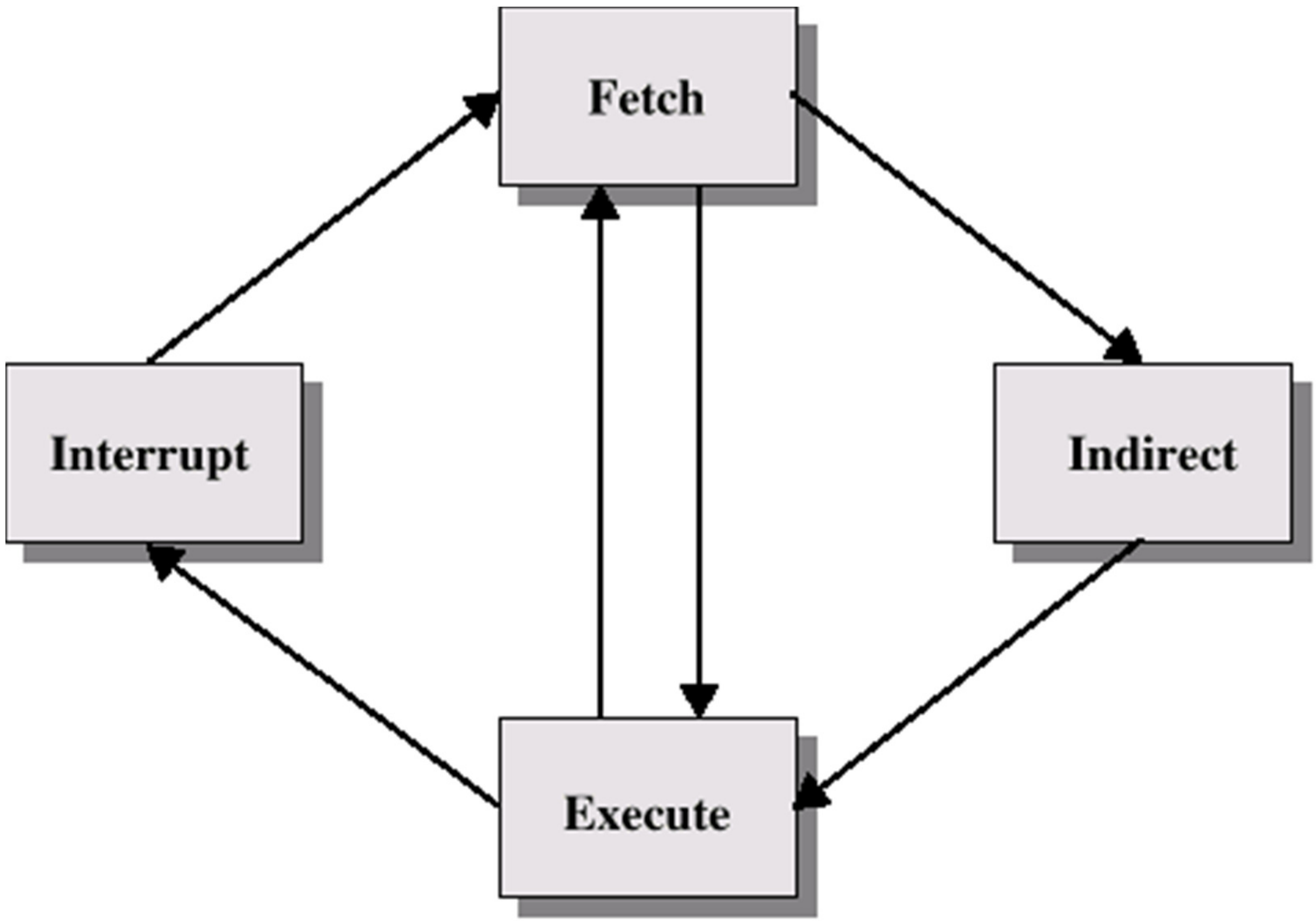
- A sequence of steps/instructions
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed

Instruction Cycle

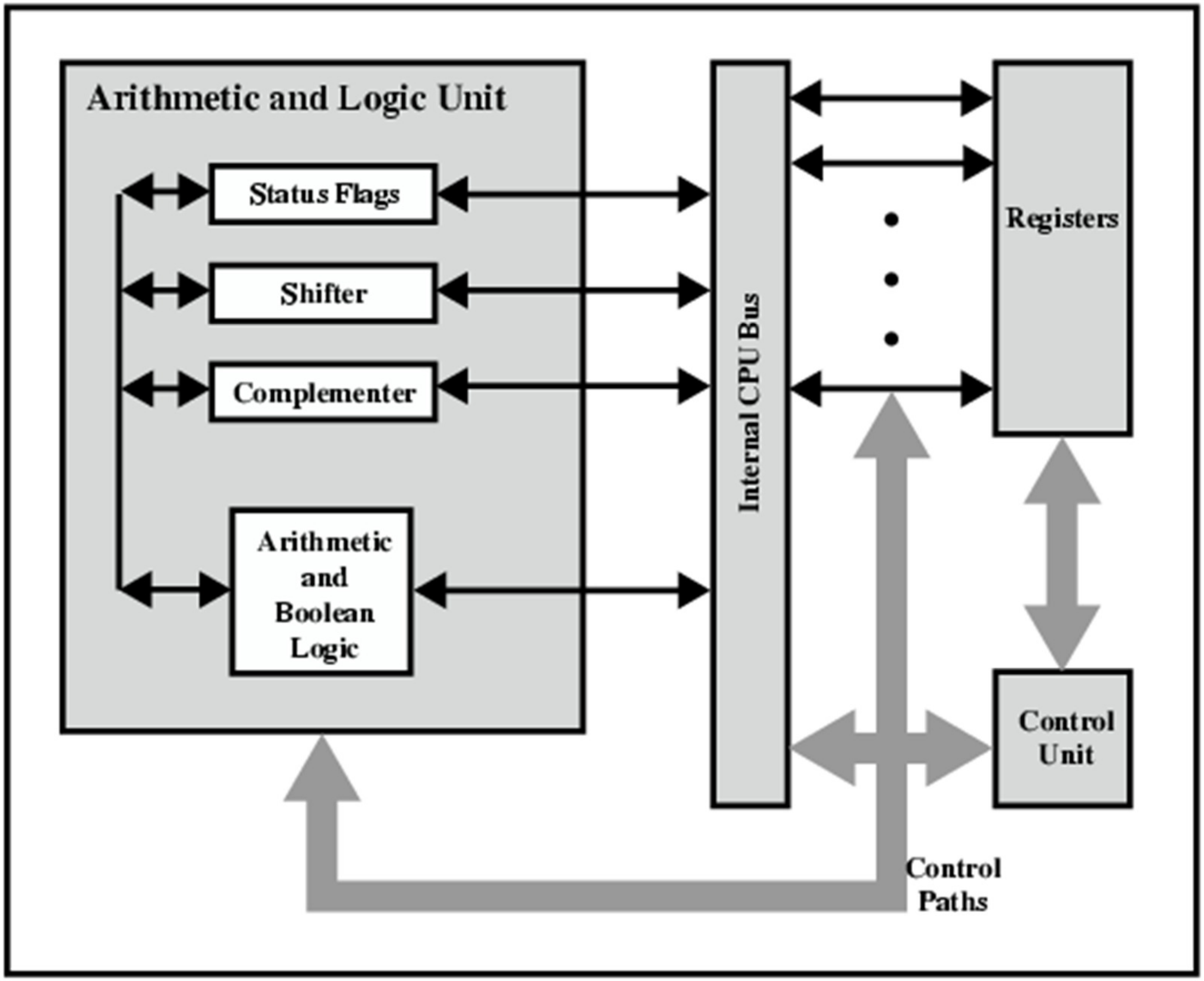
- Two steps:
  - —Fetch
  - —Execute



Instruction Cycle with Indirect



CPU Internal Structure



Registers

- CPU must have some working space (temporary storage)
- Called registers
- Number and function vary between processor designs
- One of the major design decisions
- Top level of memory hierarchy

User Visible Registers

- General Purpose
- Data
- Address
- Condition Codes

How Many GP Registers?

- Between 8 - 32
- Fewer = more memory references
- More does not reduce memory references and takes up processor space
- Large enough to hold full address
- Large enough to hold full word
- Often possible to combine two data registers
  - C programming
  - double int a;
  - long int a;

Control & Status Registers

- Program Counter
- Instruction Register
- Memory Address Register
- Memory Buffer Register

Condition Code Registers

- Sets of individual bits
  - e.g. result of last operation was zero
- Can be read (implicitly) by programs
  - e.g. Jump if zero
- Can not (usually) be set by programs
- Needs for conditional instructions

Program Status Word

- A set of bits
- Includes Condition Codes
- Sign of last result
- Zero
- Carry
- Equal
- Overflow
- Interrupt enable/disable
- Supervisor

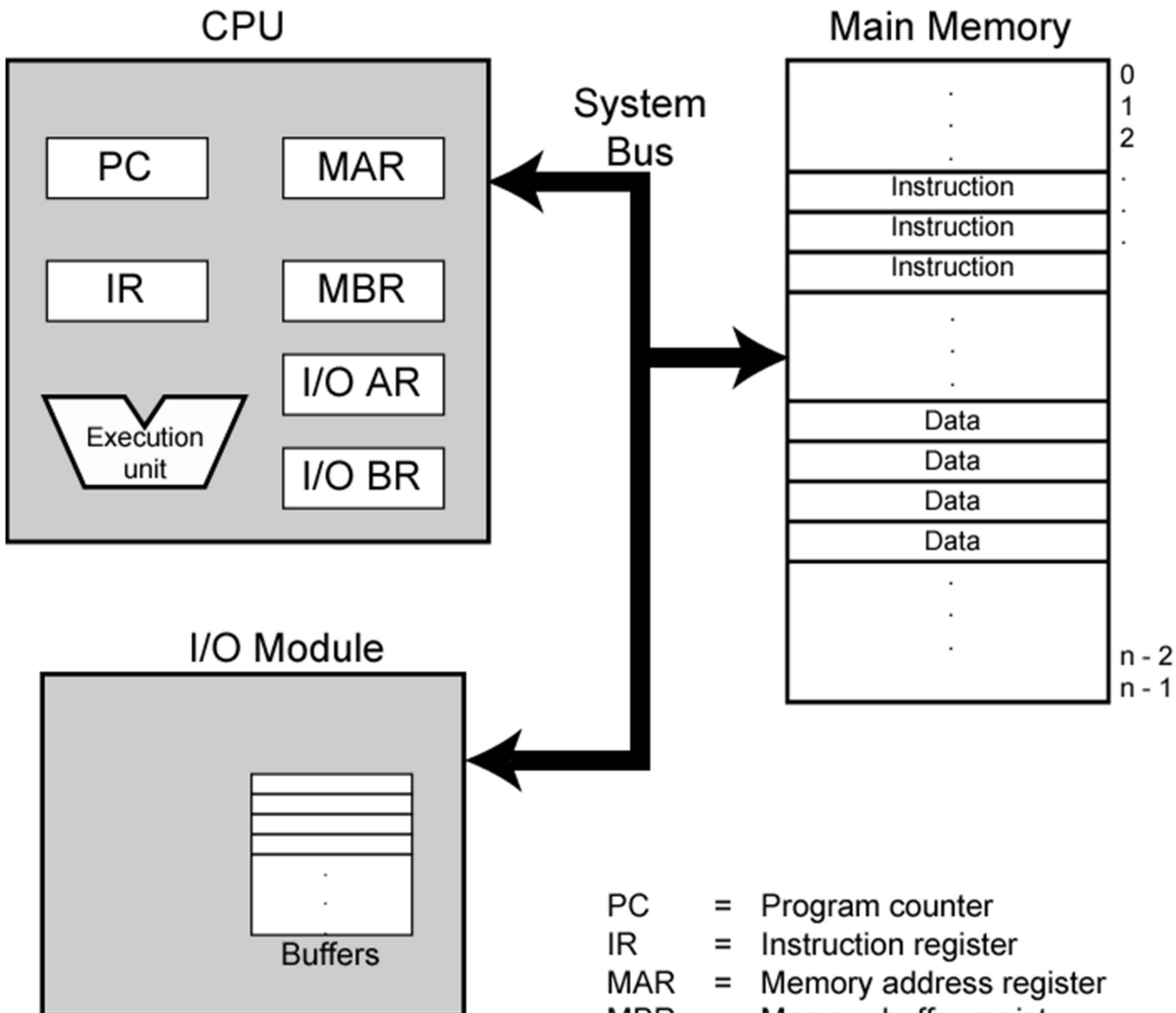
Function of Control Unit

- For each operation a unique code is provided
  - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals

Components

- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit
- Registers for internal storage
- Data and instructions need to get into the system and results out
  - Input/output
- Temporary storage of code and results is needed
  - Main memory

Computer Components: Top Level View



PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

Connecting

- All the units must be connected
- Different type of connection for different type of unit
  - Memory
  - Input/Output
  - CPU

Memory Connection

- Receives and sends data
- Receives addresses (of locations)
- Receives control signals
  - Read
  - Write
  - Timing

Input/Output Connection(1)

- Similar to memory from computer's viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer

What is a Bus?

- A communication pathway connecting two or more devices
- Usually broadcast
- Often grouped
  - A number of channels in one bus
  - e.g. 32 bit data bus is 32 separate single bit channels

Data Bus

- Carries data
  - Remember that there is no difference between “data” and “instruction” at this level
- Width is a key determinant of performance
  - 8, 16, 32, 64 bit

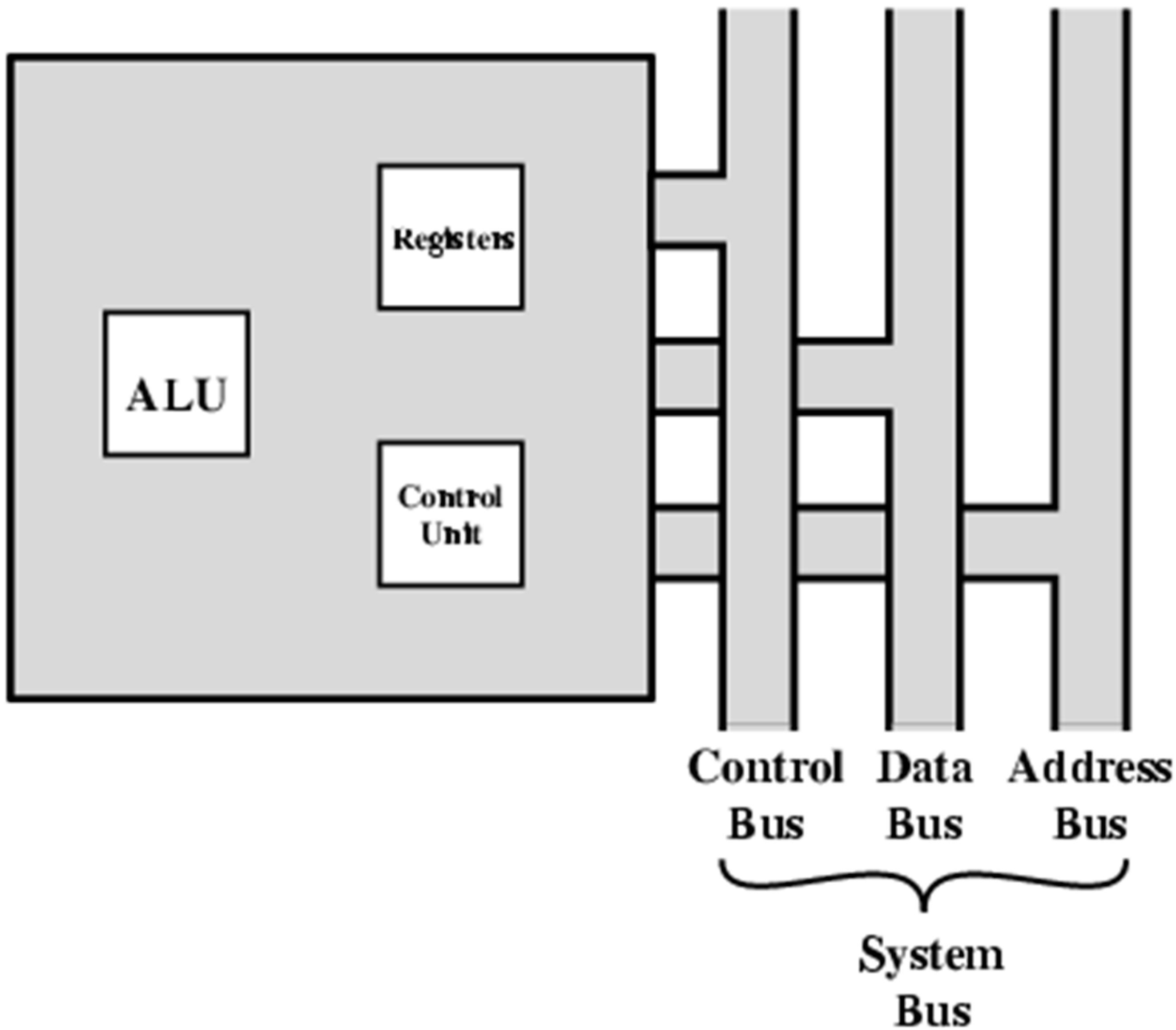
Address bus

- Identify the source or destination of data
- e.g. CPU needs to read an instruction (data) from a given location in memory
- Bus width determines maximum memory capacity of system
  - e.g. 8080 has 16 bit address bus giving 64k address space

Control Bus

- Control and timing information
  - Memory read/write signal
  - Interrupt request
  - Clock signals

CPU With Systems Bus



Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals

Program Concept

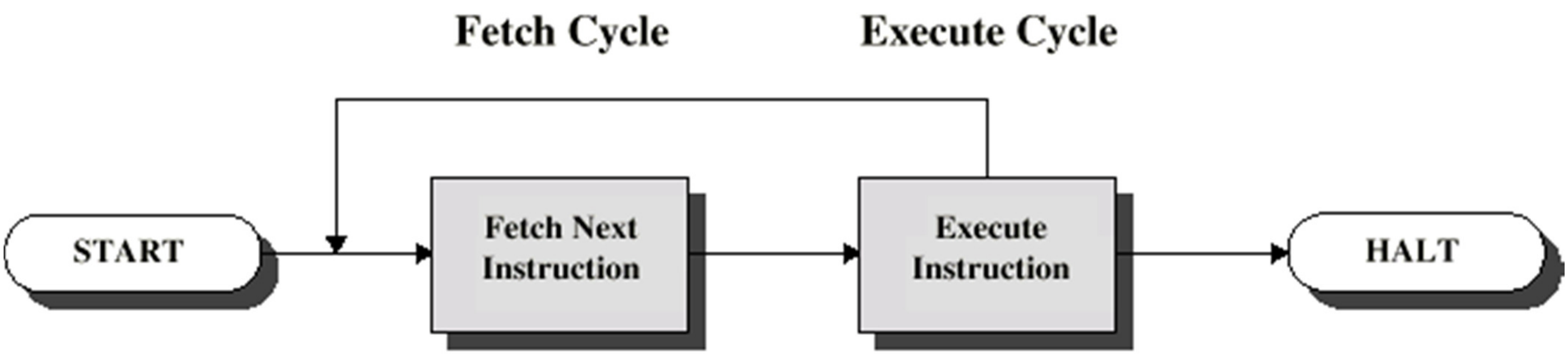
- Operating Systems
  - Viewed as an Extended Machine

What is a program?

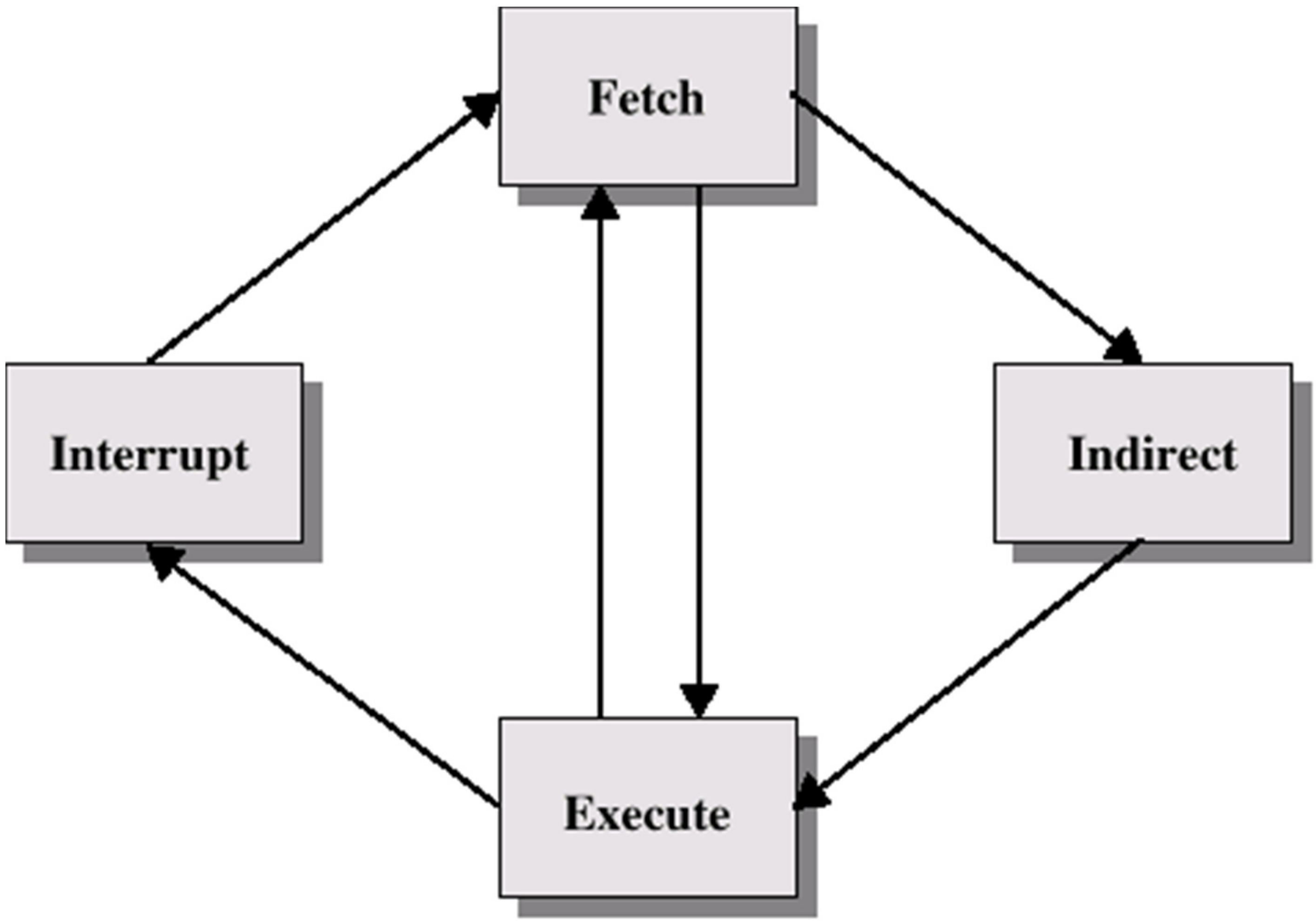
- A sequence of steps
- For each step, an arithmetic or logic operation is done
- For each operation, a different set of control signals is needed

Instruction Cycle

- Two steps:
  - Fetch
  - Execute



Instruction Cycle with Indirect



Indirect Cycle

- May require memory access to fetch operands
- Indirect addressing requires more memory accesses
- Can be thought of as additional instruction subcycle

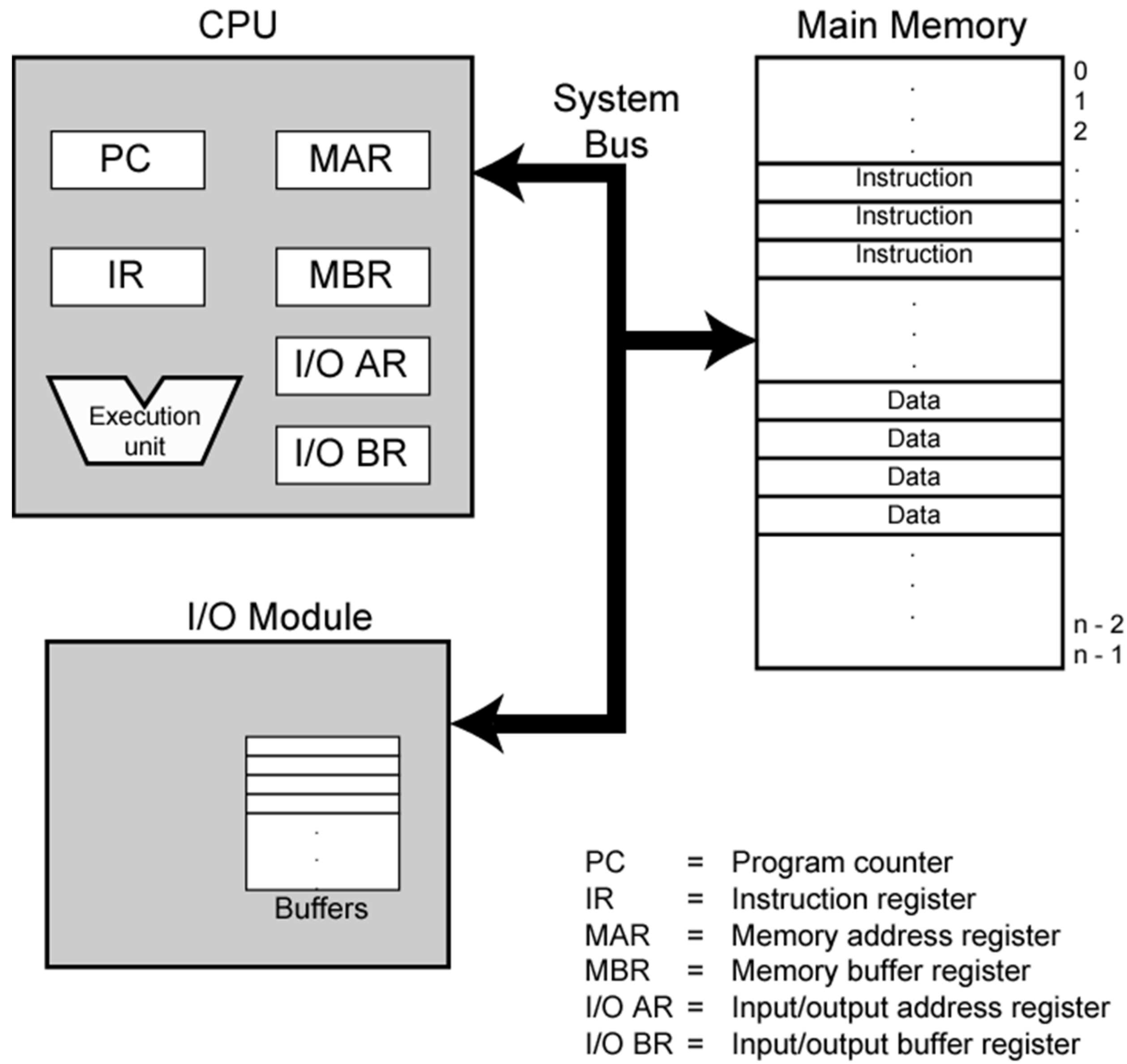
Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

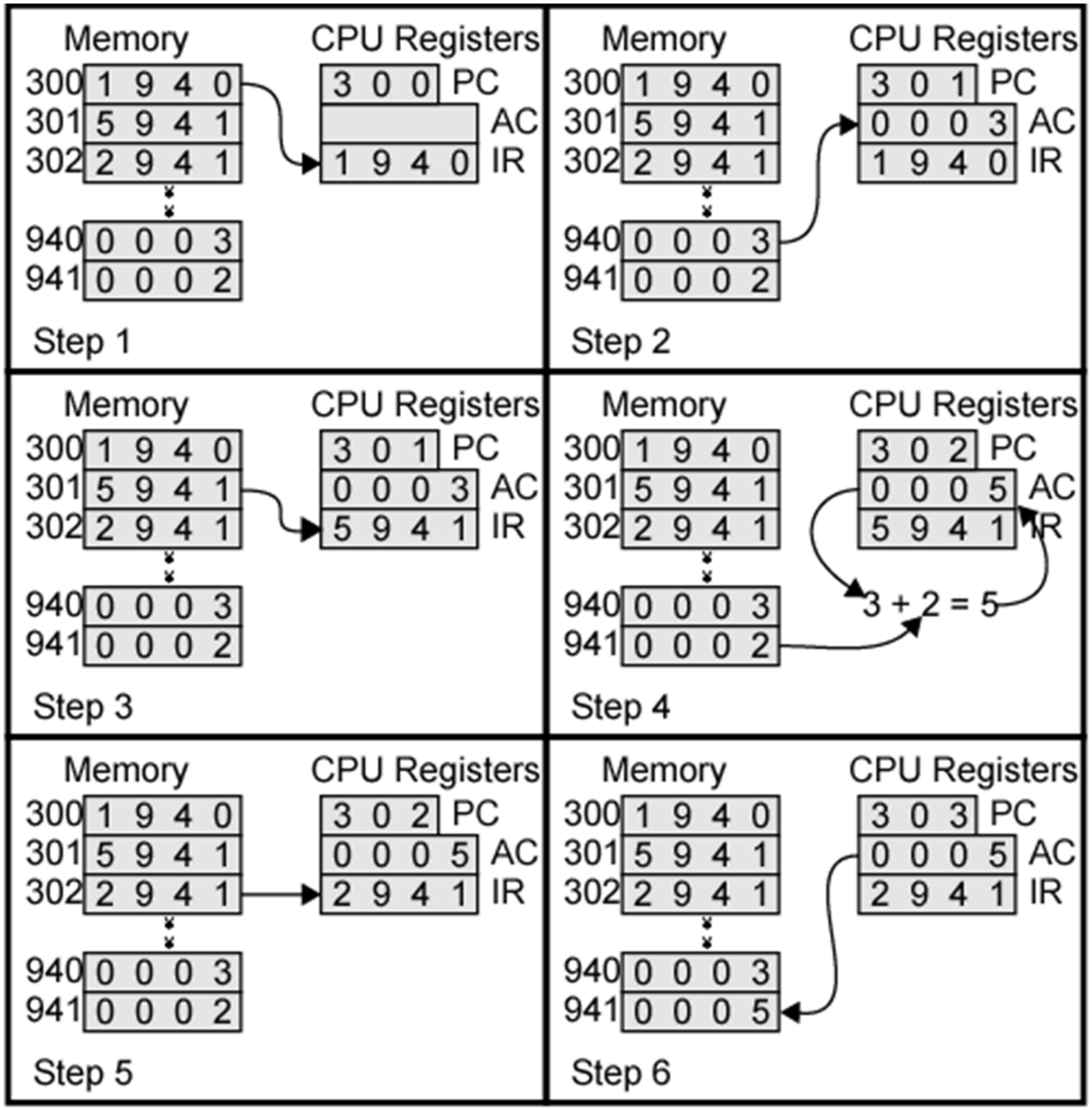
Execute Cycle

- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g. jump
- Combination of above

Computer Components: Top Level View



Example of Program Execution



Data Bus and Address Bus

- Size of Address Bus:

SIZE	BINARY	DEC	HEXA	
8	0000 0000	0	00	
8	1111 1111	255	FF	
8	0101 0111	87	57	
8	0000 0110	6	06	
10	11 1111 1111	1023	3FF	
12	1111 1111 1111	4095	FFF	
16	1111 1111 1111 1111	2^{16} -1	FFFF	
20	1111 1111 1111 1111 1111	2^{20} -1	FFFFF	
30	11 1111	2^{30} -1	3FFFFFFF	
32	1111 1111	2^{32} -1	FFFFFFFF	

Data Bus and Address Bus

- Size of Address Bus and Memory Capacity:

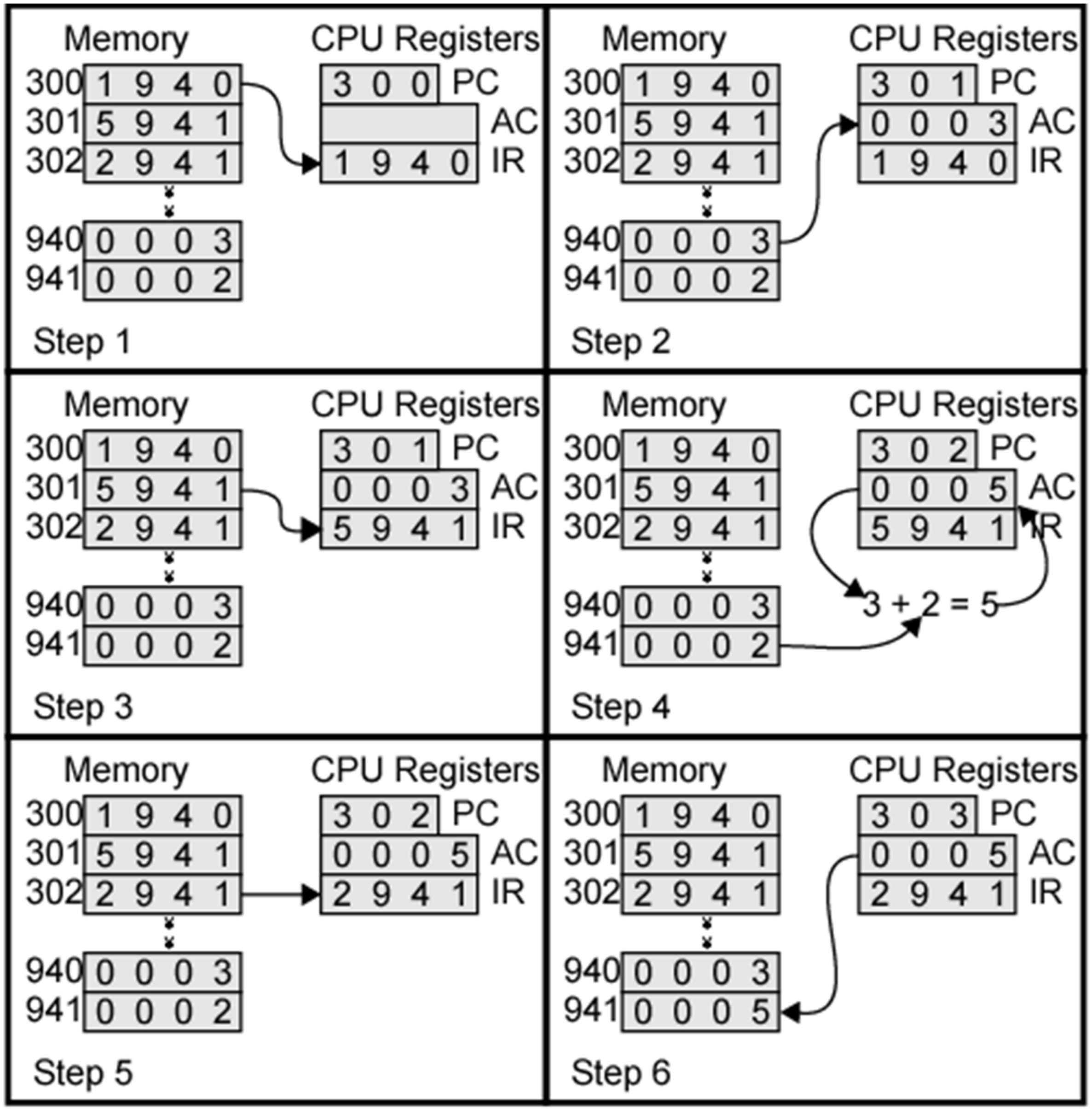
SIZE	BINARY	DEC	HEXA	
8	0000 0000	0	00	
8	1111 1111	255	FF	256
10	11 1111 1111	1023	3FF	1K
12	1111 1111 1111	4095	FFF	4K
16	1111 1111 1111 1111	2^{16} -1	FFFF	64K
20	1111 1111 1111 1111 1111	2^{20} -1	FFFFF	1M
30	11 1111	2^{30} -1	3FFFFFFF	1G
32	1111 1111	2^{32} -1	FFFFFFFF	4G

Data Bus and Address Bus

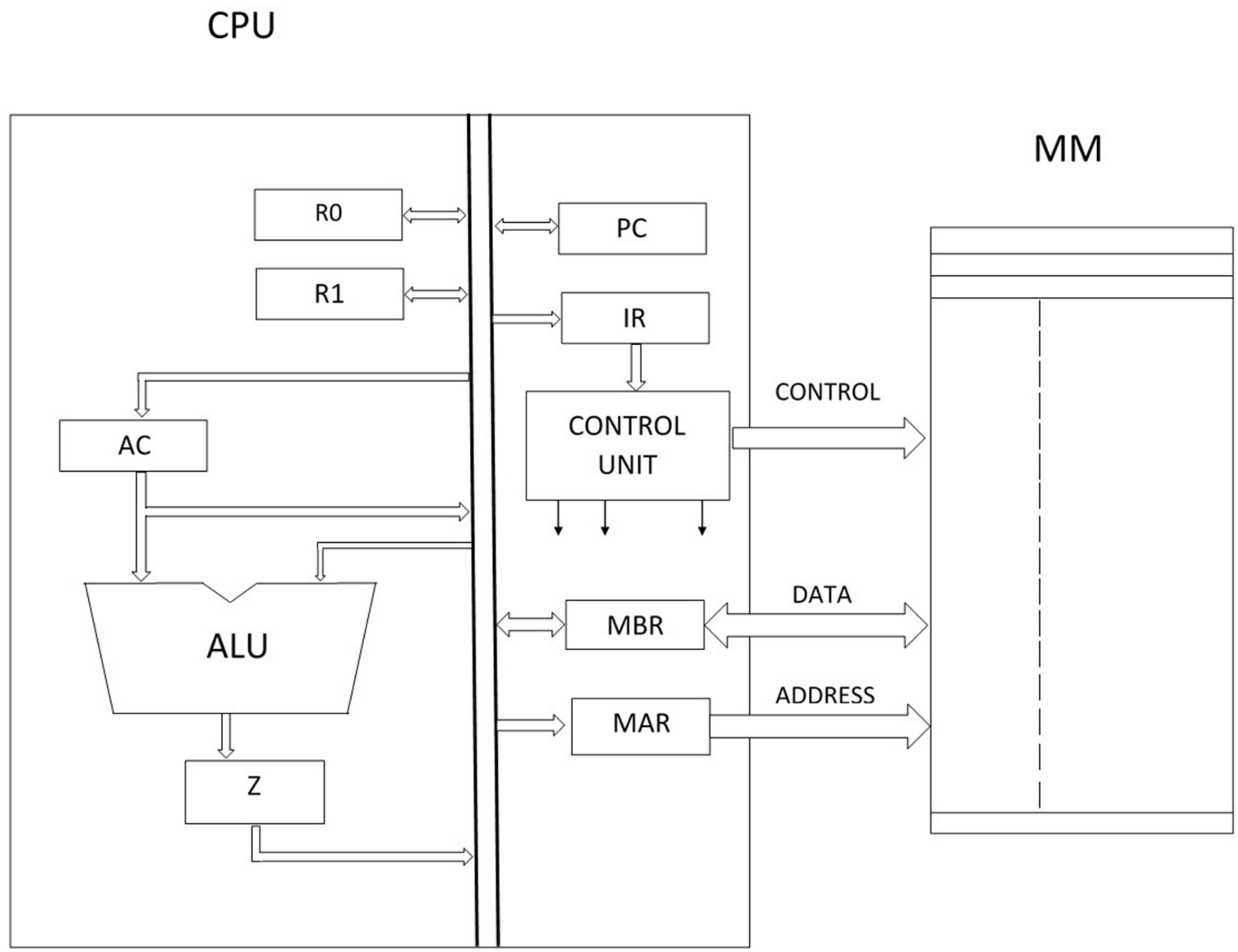
- Size of Data Bus/Memory Location:

SIZE	BINARY	DEC	HEXA
8	0000 0000 1111 1111	0 - 255	00 - FF
12	0000 0000 0000 1111 1111 1111	0 - 4095	000 - FFF
16	0000 0000 0000 0000 1111 1111 1111 1111	0 – $(2^{16} - 1)$	0000- FFFF
20	0000 0000 0000 0000 0000 1111 1111 1111 1111 1111	0 – $(2^{20} -1)$	00000 - FFFFF
32	0000 …………………………….0000 1111 …………………………….1111	0 – $(2^{32} -1)$	00000000 – FFFFFFFF

Example of Program Execution



CPU Organization



Machine Instruction

Machine	Instruction Format				Assembly
Instruction	Operation	Address			Code
1940	0001	1001	0100	0000	LDA M
5941	0101	1001	0100	0001	ADD M
2941	0010	1001	0100	0001	STA M

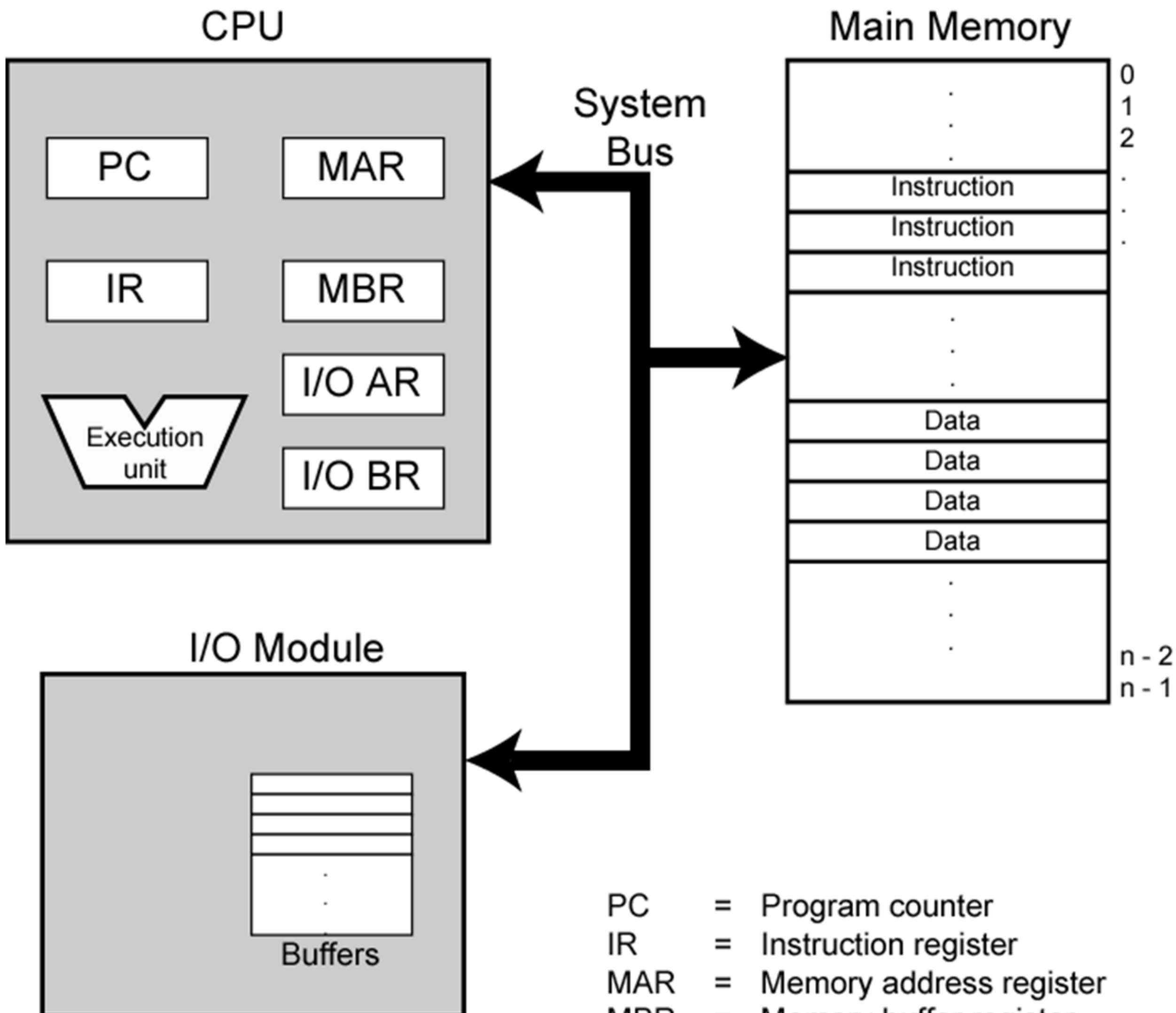
(LDA M) LOAD AC: Load the accumulator by the contents of memory location specified in the instruction

(ADD M) ADD AC: Add the contents of memory location specified in the instruction to accumulator and store the result in accumulator

(STA M) STORE AC: Store the contents of accumulator the memory location specified in the instruction

High Level Code	Assembly Code	Machine Code (HEX)
Y = X + Y	LDA X ADD Y STA Y	1940 5941 2941

Computer Components: Top Level View

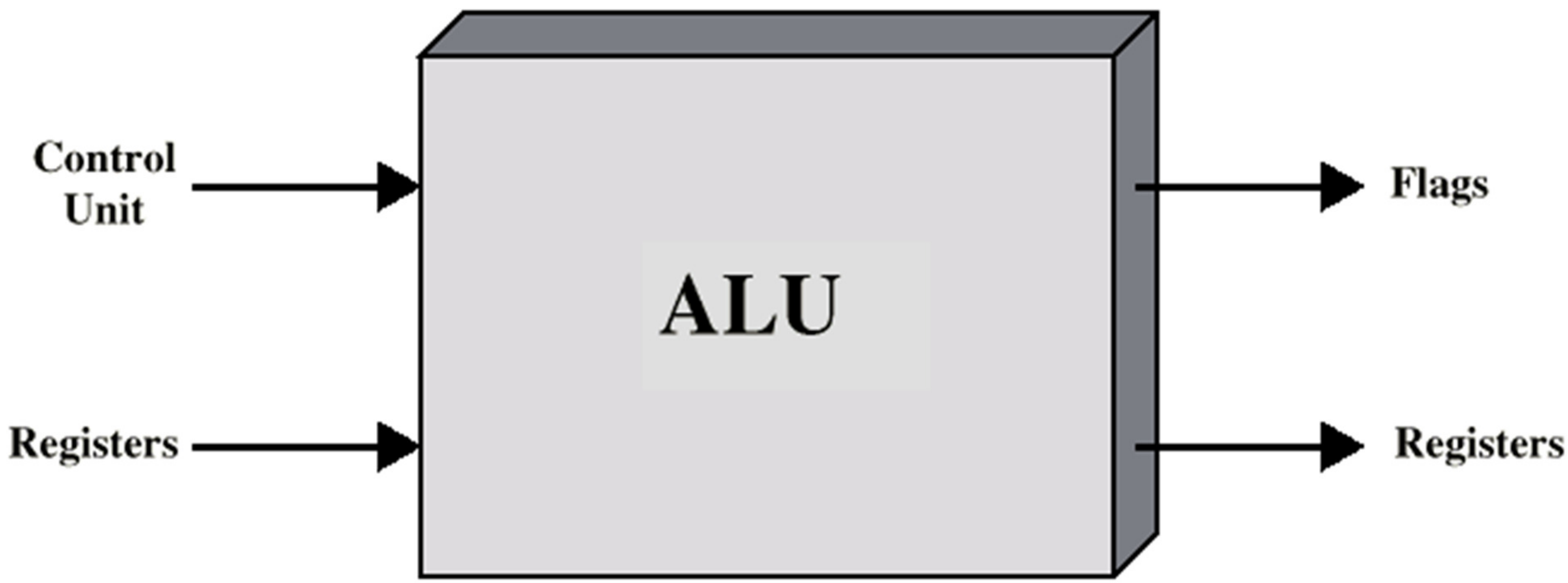


PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

Arithmetic & Logic Unit

- Does the calculations
- Handles integers
- May handle floating point (real) numbers
- May be separate FPU (maths co-processor)
- May be on chip separate FPU

ALU Inputs and Outputs



Integer Representation

- Only have 0 & 1 to represent everything
- numbers stored in binary
  - e.g. 41=00101001
- No minus sign (negative nos.)
- No period (for real nos.)
- Negative Numbers
  - Sign-Magnitude
  - Two’s compliment

Sign-Magnitude

- Left most bit is sign bit (MSB)
  - 0 means positive
  - 1 means negative
- +18 = 00010010
- -18 = 10010010
- Problems
  - Need to consider both sign and magnitude in arithmetic
  - Two representations of zero (+0 and -0)

Two's Compliment

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Benefits

- One representation of zero
- Arithmetic works easily
- Negating is fairly easy
  - 3 = 00000011
  - Boolean complement gives 11111100
  - Add 1 to LSB 11111101

Negation Special Case 1

- 0 = 00000000
- Bitwise not 11111111
- Add 1 to LSB +1
- Result 1 00000000
- Overflow is ignored, so:
- - 0 = 0 √

Negation Special Case 2

- -128 = 10000000
- bitwise not 01111111
- Add 1 to LSB +1
- Result 10000000
- So:
- -(-128) = -128 X
- Monitor MSB (sign bit)
- It should change during negation

Range of Numbers

- 8 bit 2s compliment
  - +127 = 01111111 = 2^7 -1
  - -128 = 10000000 = -2^7
- 16 bit 2s compliment
  - +32767 = 011111111 11111111 = 2^{15} - 1
  - -32768 = 100000000 00000000 = -2^{15}

Real Numbers

- Numbers with fractions
- Could be done in pure binary
  - 1001.1010 = $2^4 + 2^0 + 2^{-1} + 2^{-3}$ = 9.625
- Where is the binary point?
- Fixed?
  - Very limited
- Moving?
  - How do you show where it is?

Floating Point

(a) Format

- +/- .significand x $2^{exponent}$
- Misnomer
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

Floating Point Examples

(a) Format

$$
\begin{array}{rcl}
1.1010001 \times 2^{10100} & = 0\ 10001001\ 10100010000000000000000 = & 1.638125 \times 2^{20} \\
-1.1010001 \times 2^{10100} & = 1\ 10001001\ 10100010000000000000000 = & -1.638125 \times 2^{20} \\
1.1010001 \times 2^{-10100} & = 0\ 01101011\ 10100010000000000000000 = & 1.638125 \times 2^{-20} \\
-1.1010001 \times 2^{-10100} & = 1\ 01101011\ 10100010000000000000000 = & -1.638125 \times 2^{-20}
\end{array}
$$

(b) Examples

Signs for Floating Point

- Mantissa is stored in 2s compliment
- Exponent is in excess or biased notation
  - e.g. Excess (bias) 128 means
  - 8 bit exponent field
  - Pure value range 0-255
  - Subtract 128 to get correct value
  - Range -128 to +127

Normalization

- FP numbers are usually normalized
- i.e. exponent is adjusted so that leading bit (MSB) of mantissa is 1
- Since it is always 1 there is no need to store it
- (c.f. Scientific notation where numbers are normalized to give a single digit before the decimal point
- e.g. 3.123×10^3)

FP Ranges

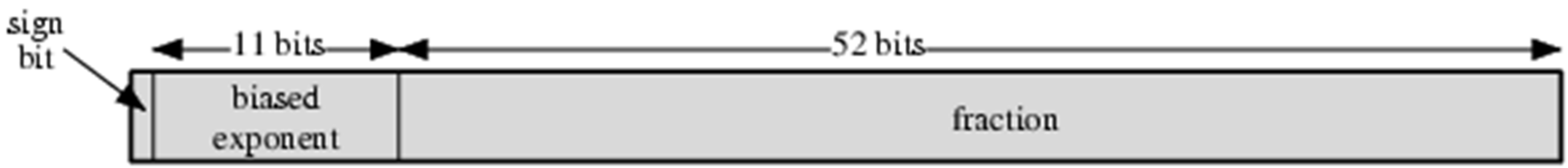
- For a 32 bit number
  - 8 bit exponent
  - +/- $2^{256} \approx 1.5 \times 10^{77}$
- Accuracy
  - The effect of changing lsb of mantissa
  - 23 bit mantissa $2^{-23} \approx 1.2 \times 10^{-7}$
  - About 6 decimal places

IEEE 754

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively

IEEE 754 Formats

(a) Single format



(b) Double format

Other Codes

- Excess Code (Excess-128)
- GREY Code
- BCD (Binary Coded Decimal)

Character Representation

- ASCII (American Standard Code for Information Interchange)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- UNICODE