System and Devices (SYS 3)

Lecture 3: Computer Organization Basics Review

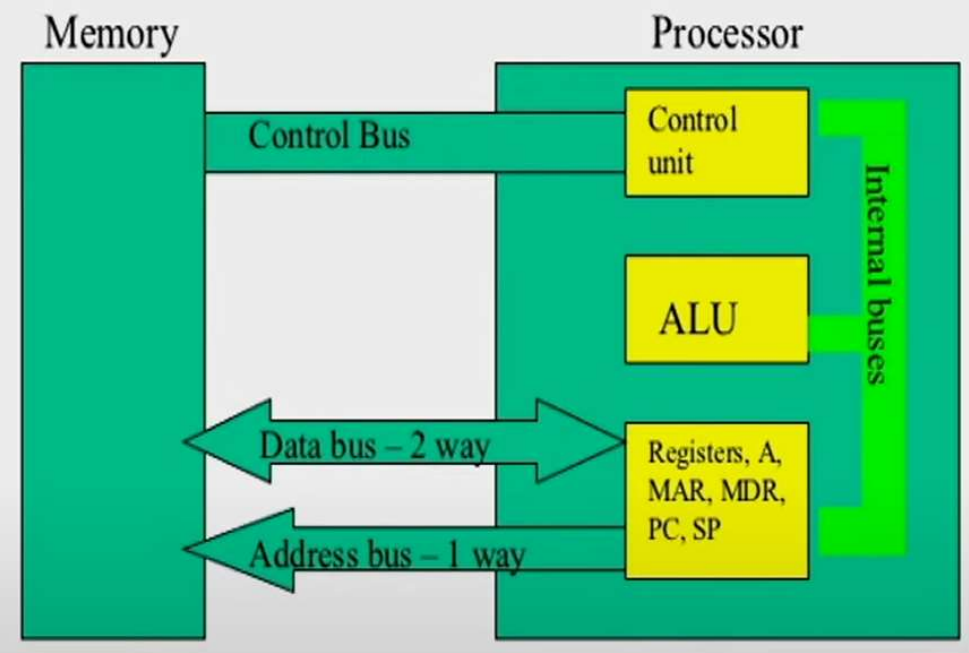
Dr Pengcheng Liu

Instruction Execution Cycle

Instruction Fetch	Obtain instruction from program storage
Instruction Decode	Determine required actions and instruction size
Operand Fetch	Locate and obtain operand data
Execute	Compute result value or status
Result Store	Deposit results in storage for later use
Next Instruction	Determine successor instruction

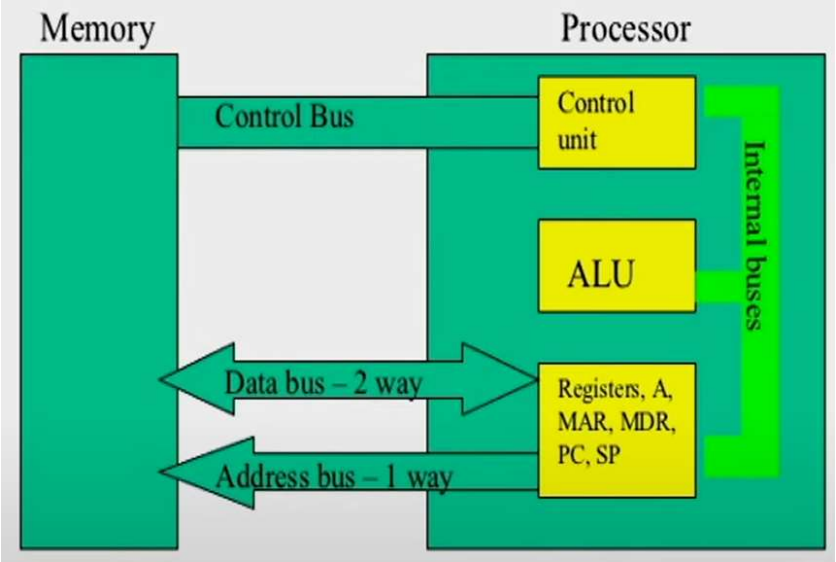
Processor Memory interaction

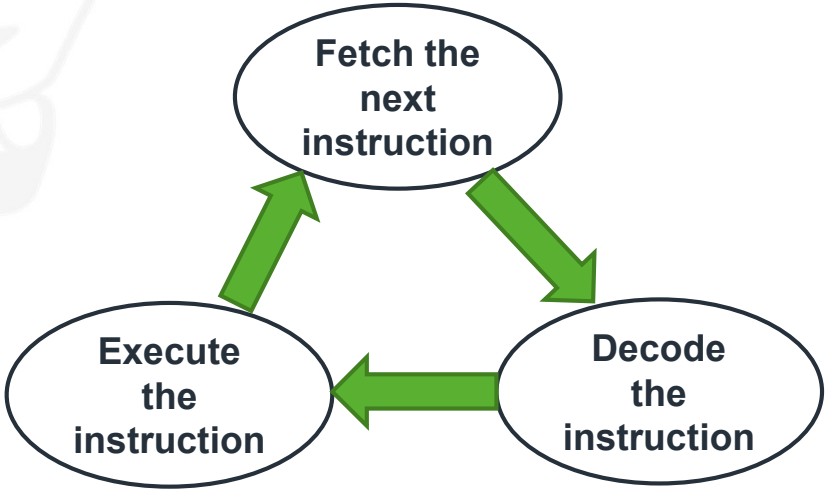
- Instruction Fetch
- Instruction Decode
- Operand Fetch
- Execute
- Result Store
- Next Instruction

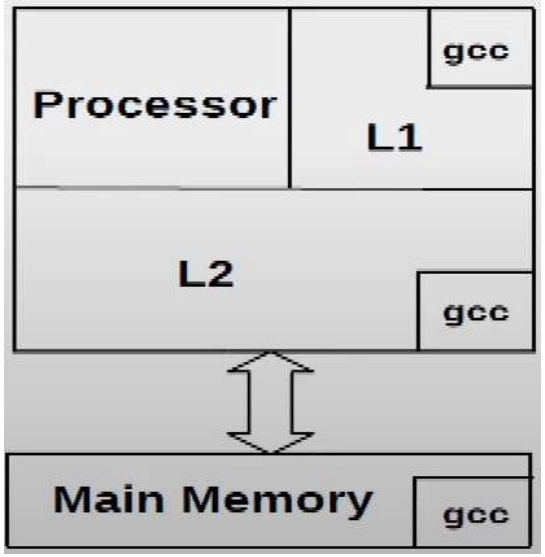


Processor Memory interaction

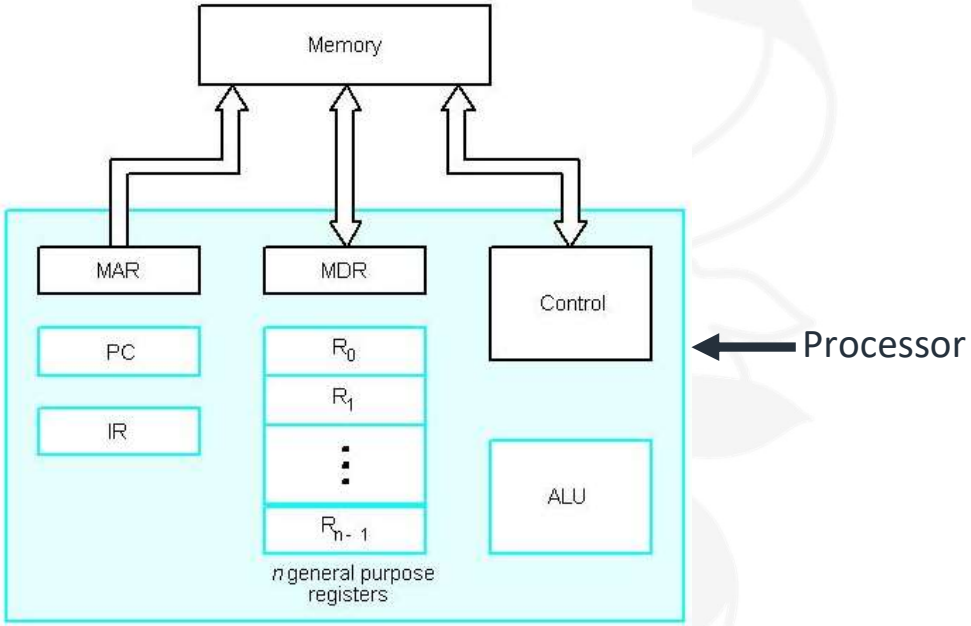
- Instruction Fetch
- Instruction Decode
- Operand Fetch
- Execute
- Result Store
- Next Instruction

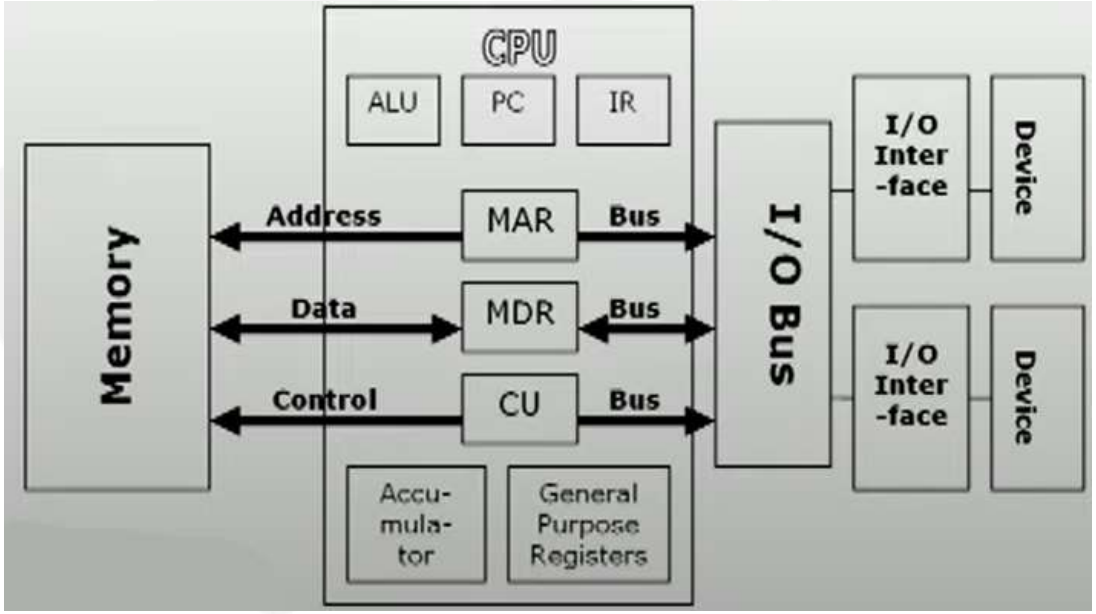


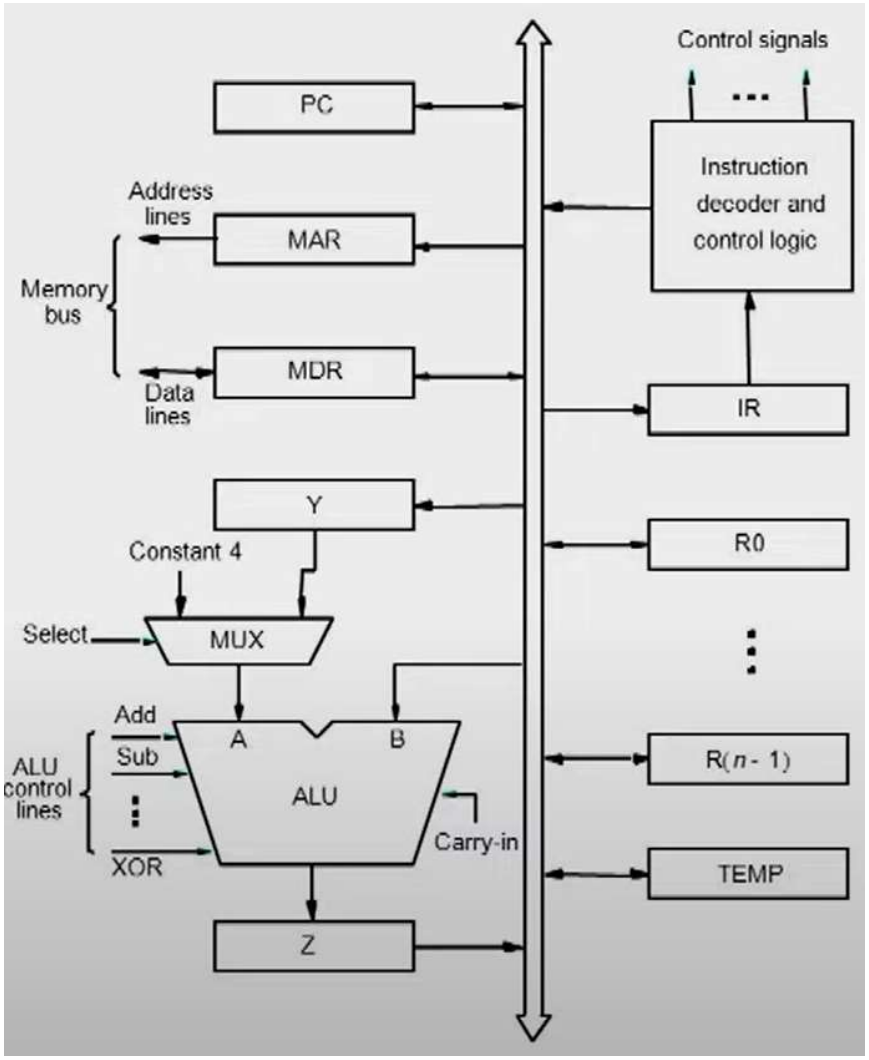




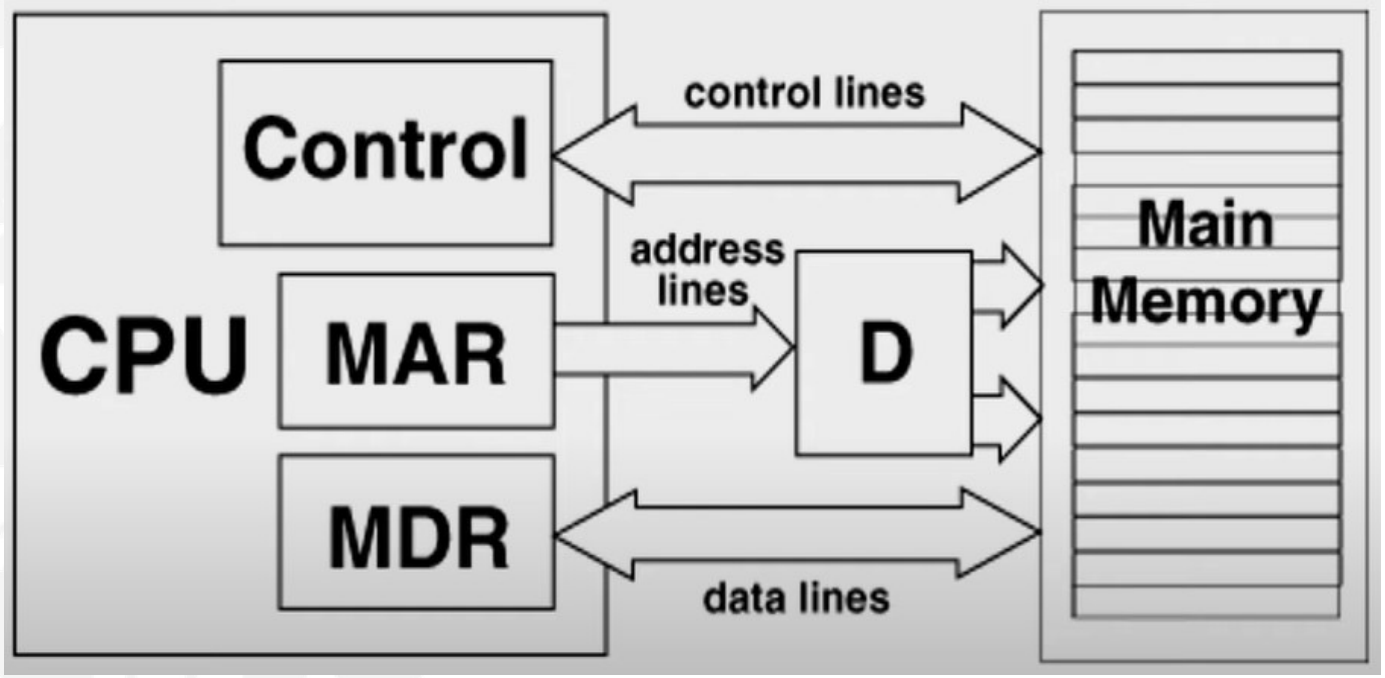
Processor Memory interaction



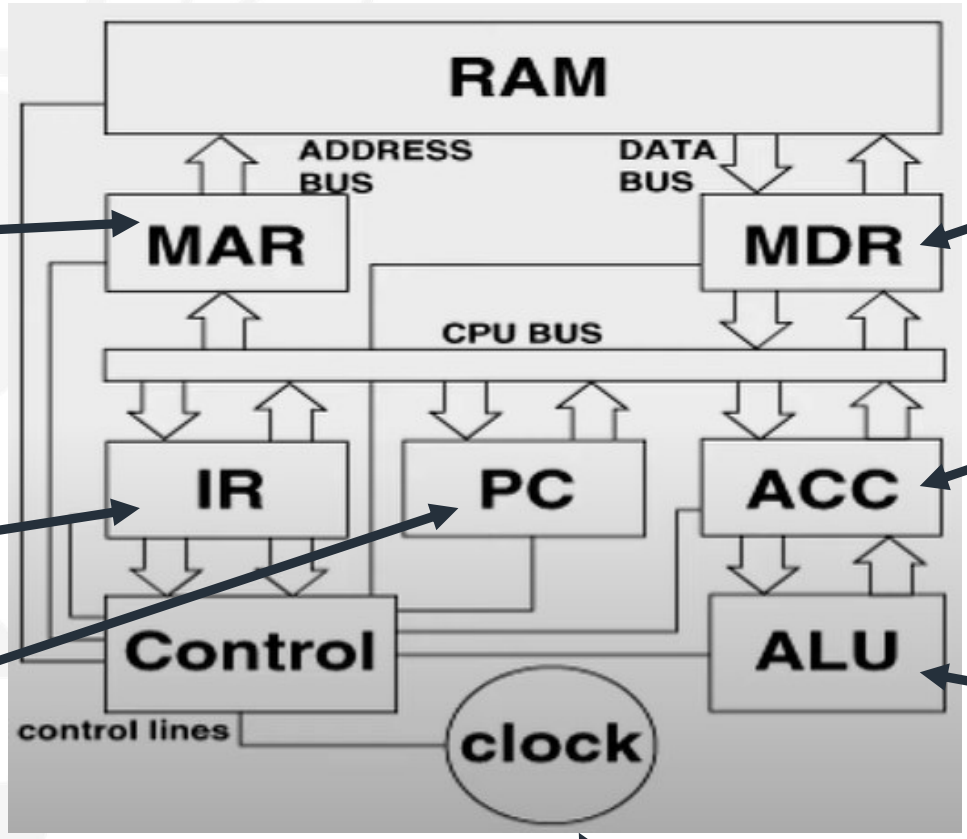




Processor Memory interaction



Inside the CPU



Memory address register

Instruction register

Program counter

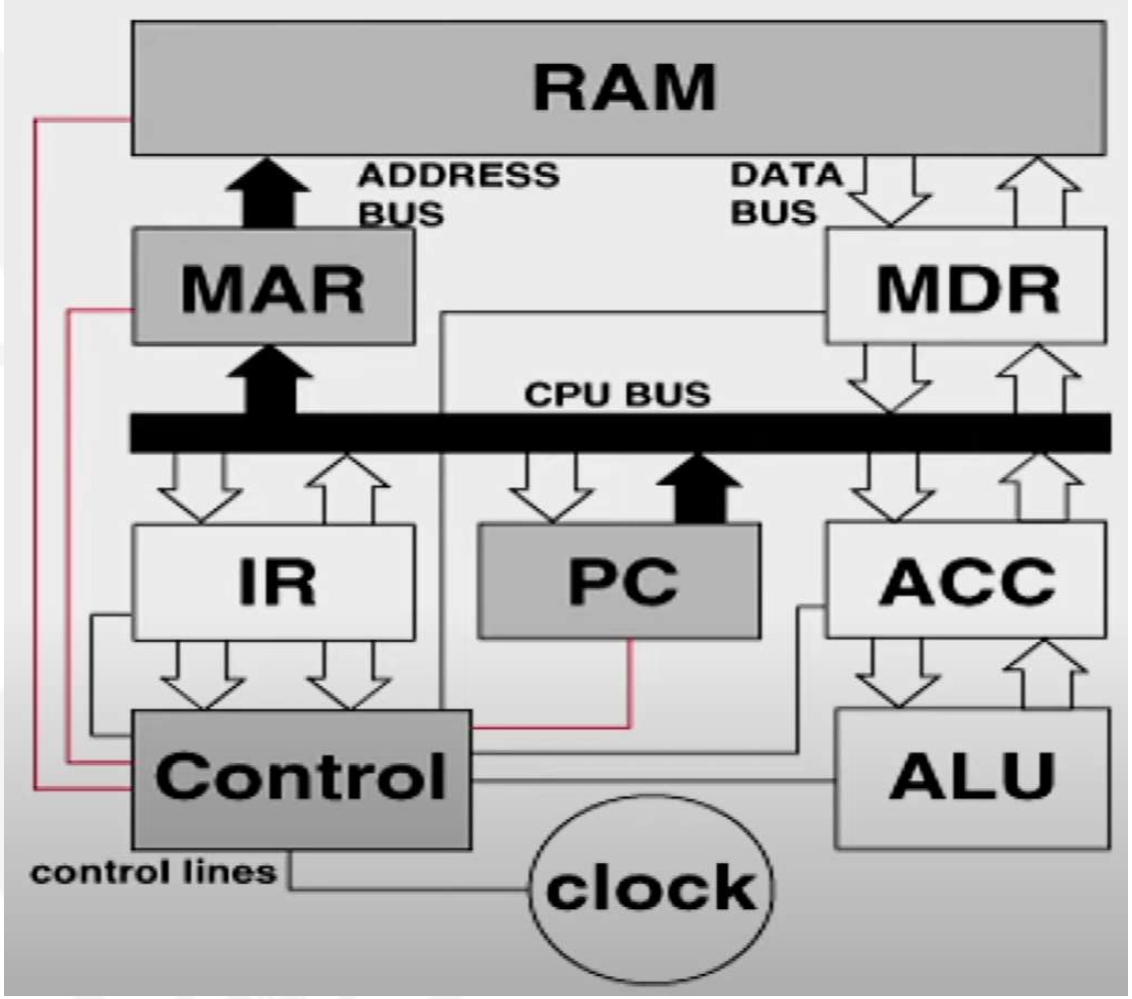
Memory data register

Accumulator (work register)

Arithmetic logic unit

System clock

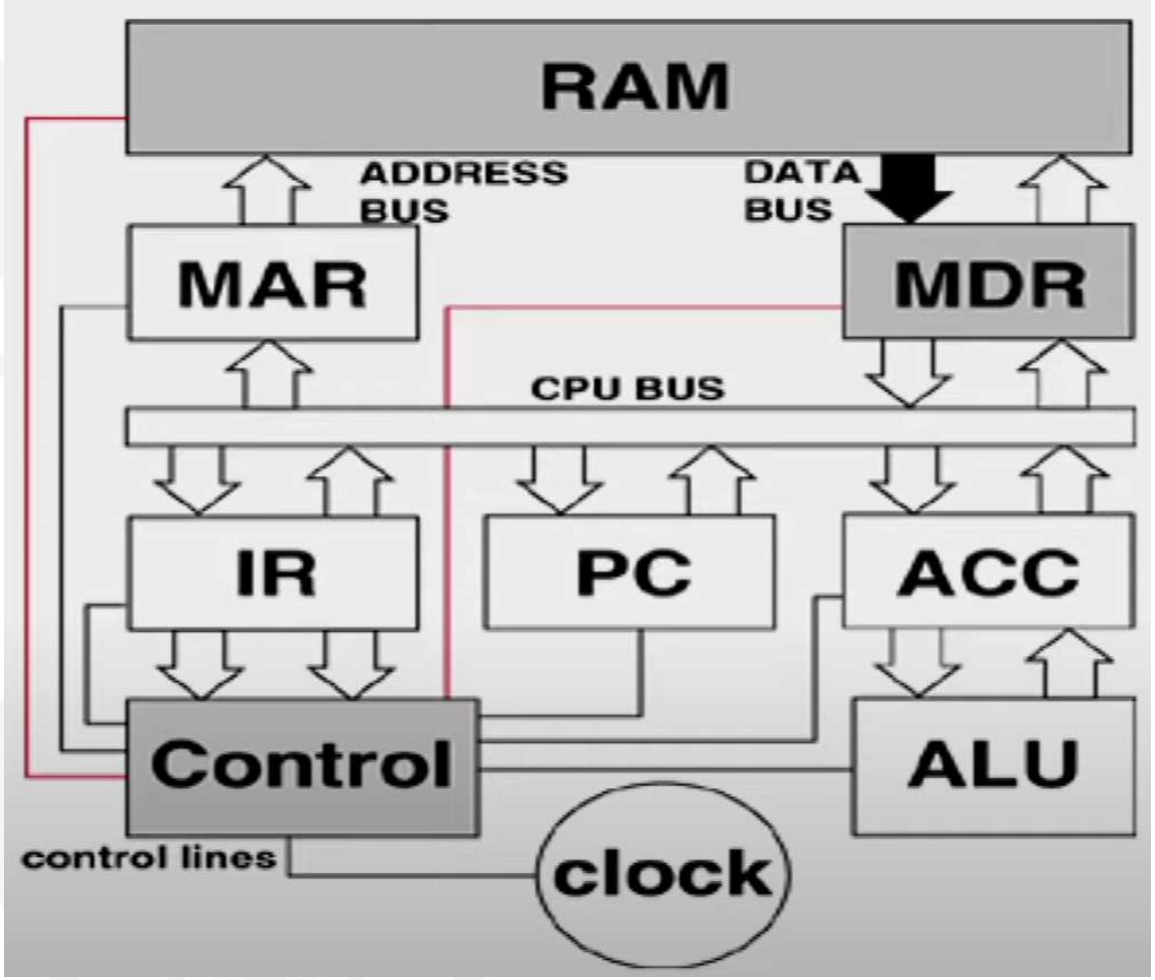
Fetch the instruction



1. Address of the next instruction is transferred from PC to MAR.

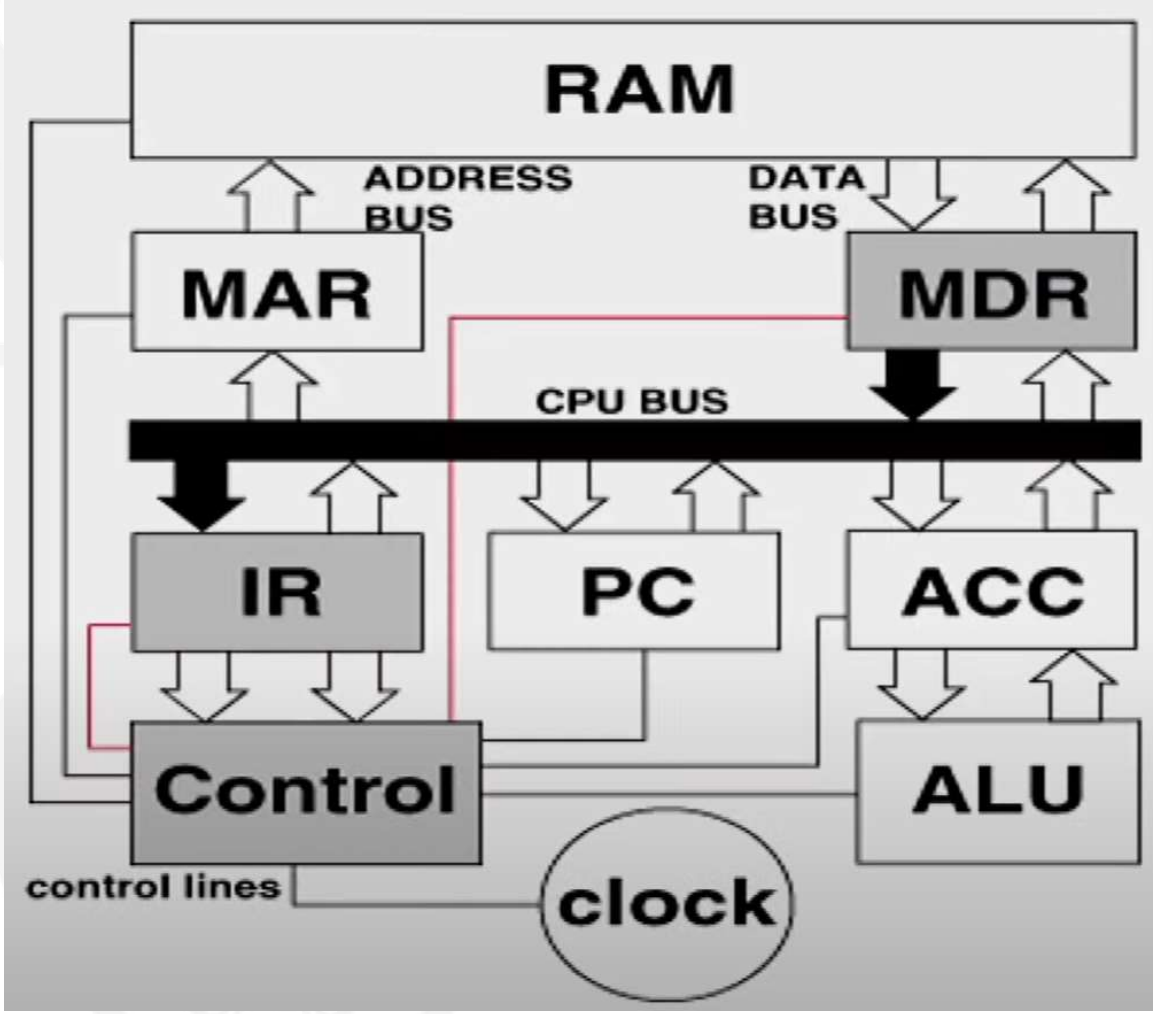
2. The instruction is located in memory.

Fetch the instruction



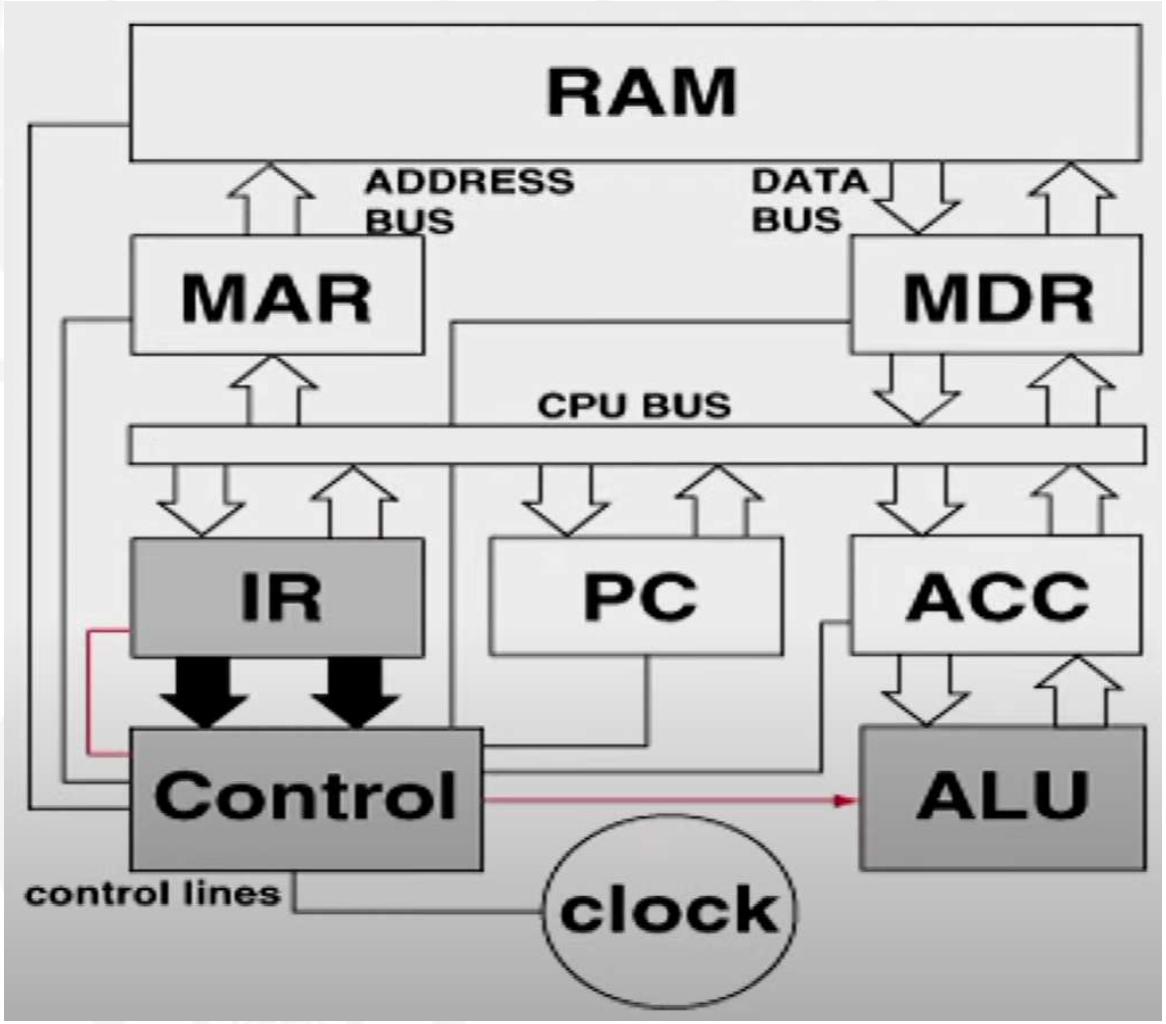
3. Instruction is copied from memory to MDR.

Decode the instruction



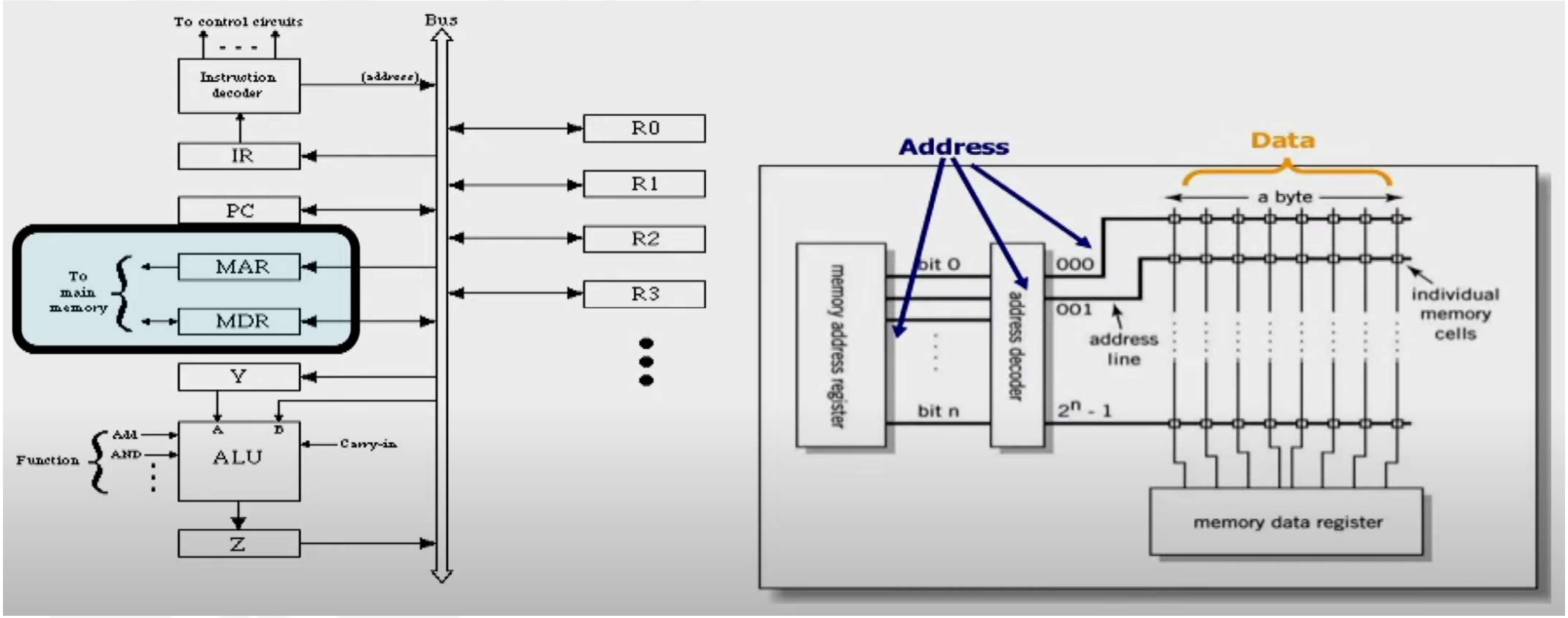
4. Instruction is transferred to and decoded in the IR.

Execute the instruction

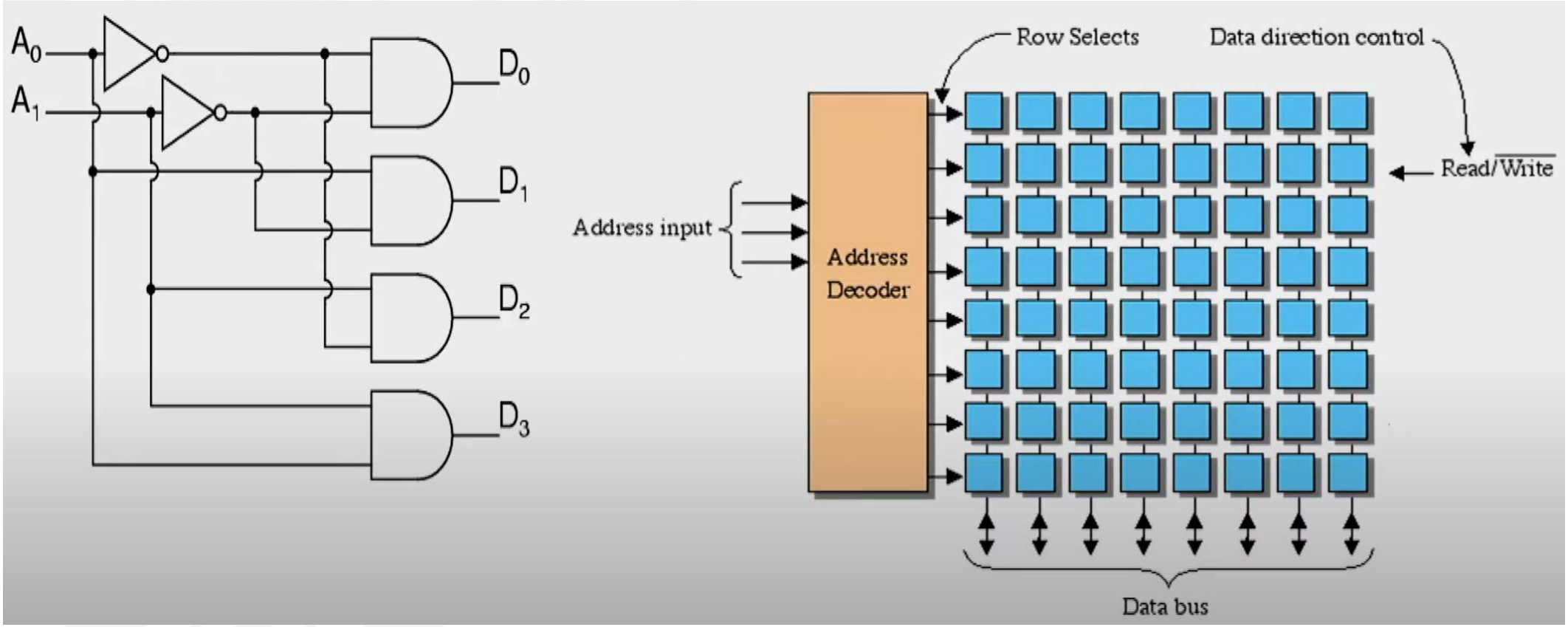


5. Control unit sends signals to appropriate devices to cause execution of the instruction.

Processor Memory Interaction



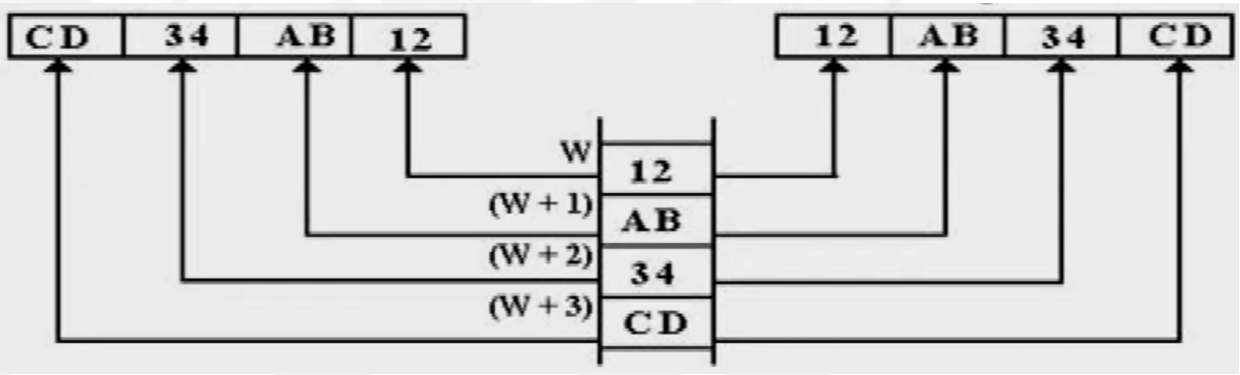
How memory works?



Byte Ordering
Big Endian VS Little Endian

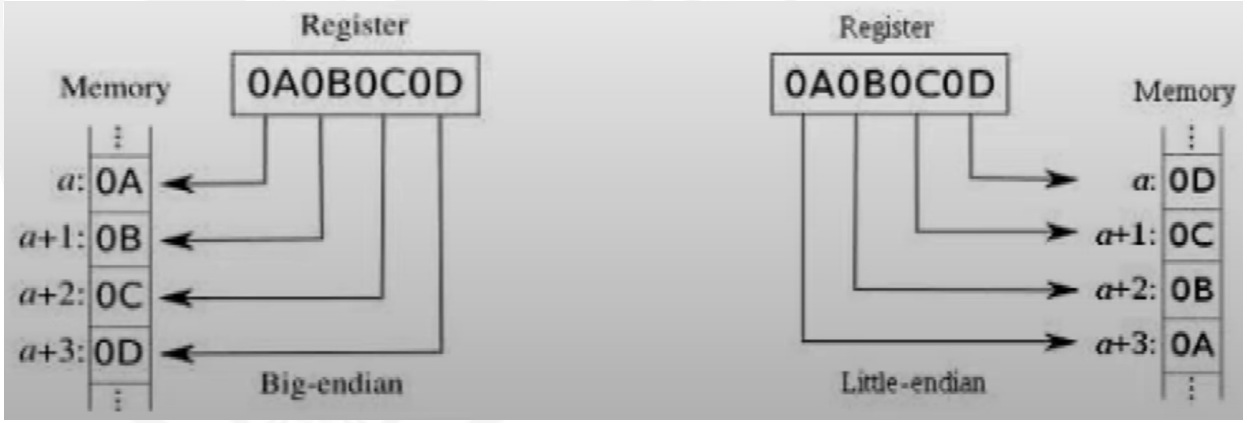
Little Endian

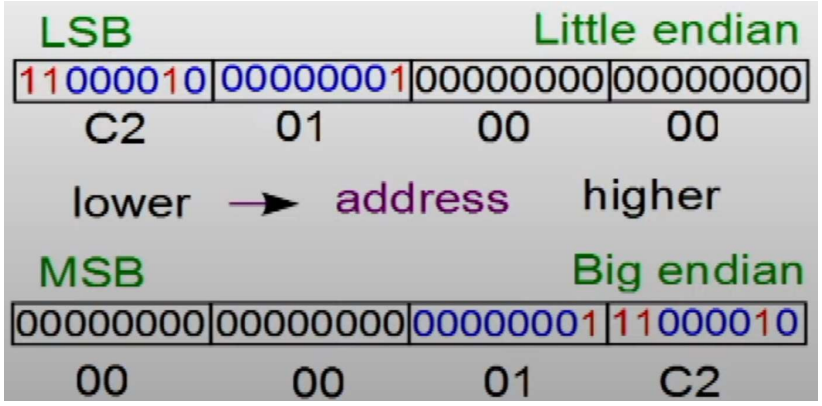
Big Endian



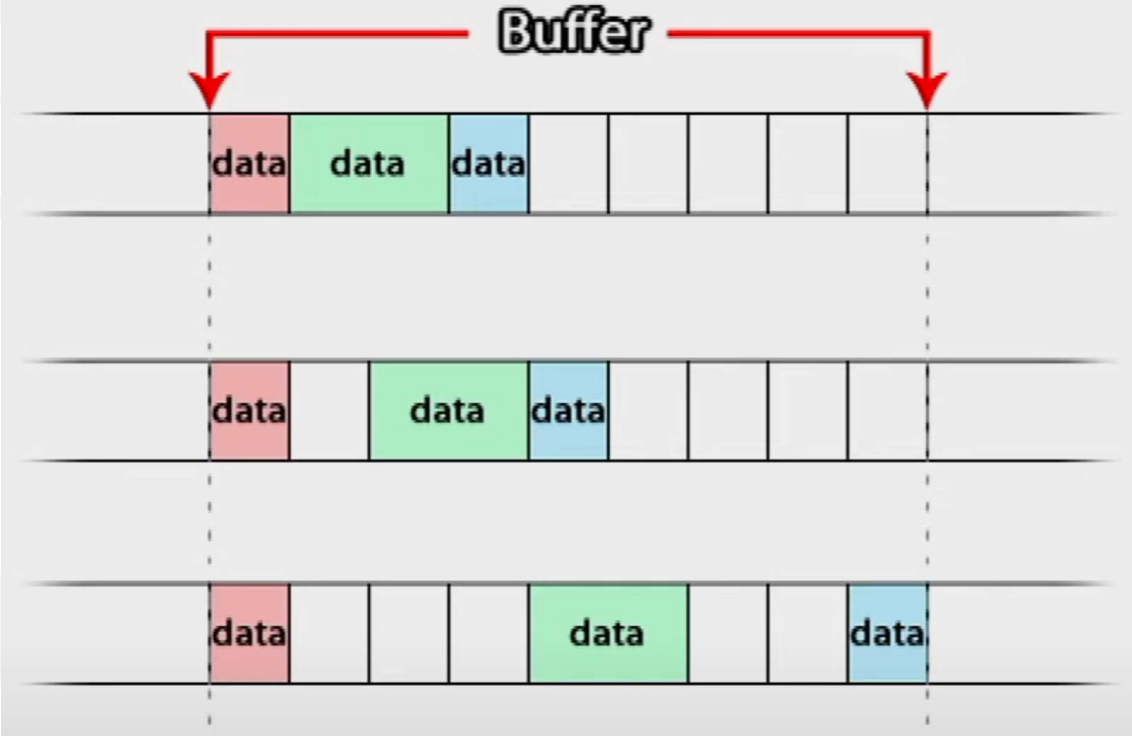
32=bit number:
0XCD34AB12

Int i = 450 = $2^8+2^7+2^6+2$ =
X000001C2





Byte / Word Alignment

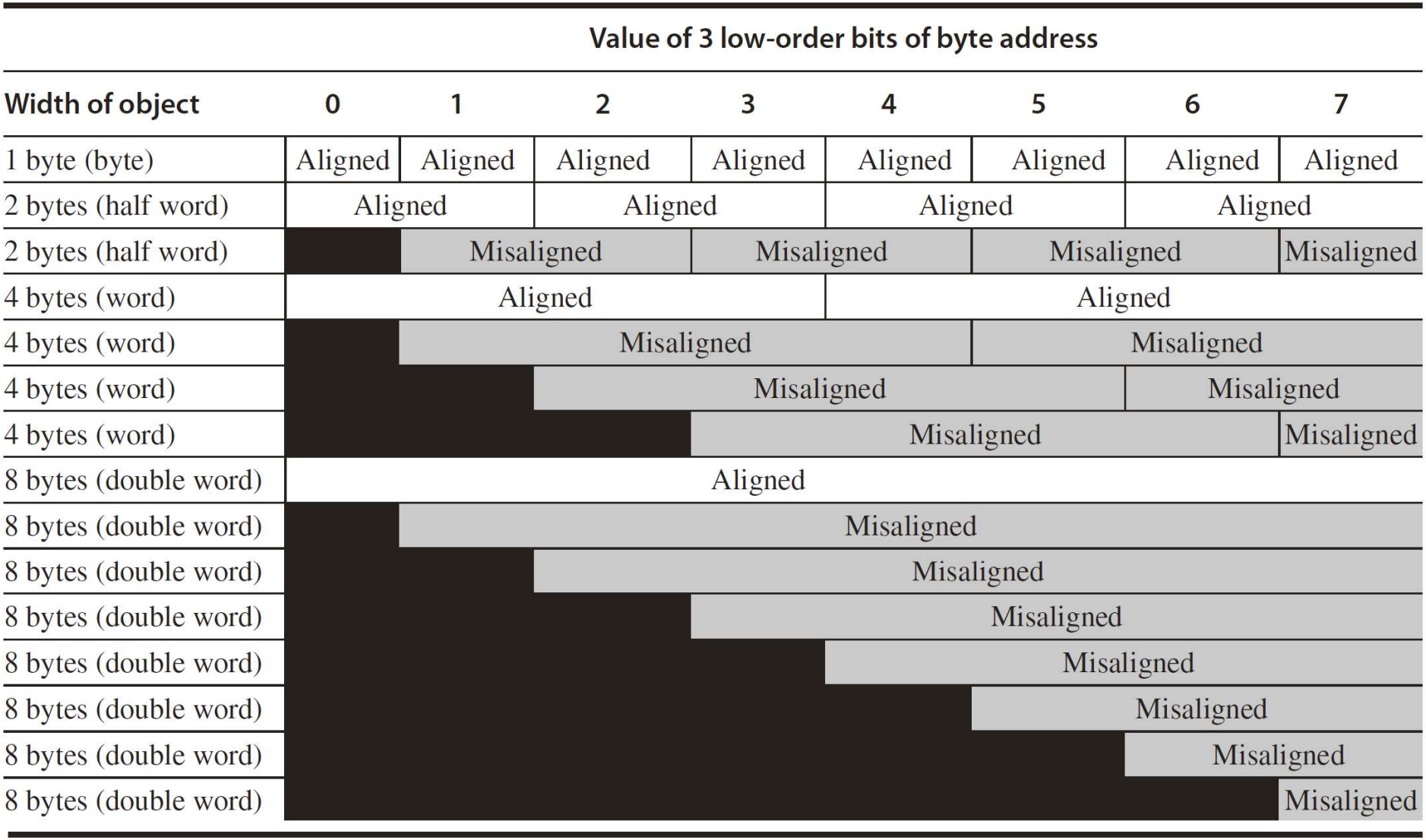


Aligned to 1 byte

Aligned to 2 bytes

Aligned to 4 bytes

Byte / Word Alignment



	Value of 3 low-order bits of byte address							
Width of object	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1 byte (byte)	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned
2 bytes (half word)	Aligned		Aligned		Aligned		Aligned	
2 bytes (half word)		Misaligned		Misaligned		Misaligned		Misaligned
4 bytes (word)	Aligned				Aligned			
4 bytes (word)		Misaligned				Misaligned		
4 bytes (word)			Misaligned				Misaligned	
4 bytes (word)				Misaligned				Misaligned
8 bytes (double word)	Aligned							
8 bytes (double word)		Misaligned						
8 bytes (double word)			Misaligned					
8 bytes (double word)				Misaligned				
8 bytes (double word)					Misaligned			
8 bytes (double word)						Misaligned		
8 bytes (double word)							Misaligned	
8 bytes (double word)								Misaligned

Types of Processor Operations

Arithmetic and Logical Operations

- Integer arithmetic
- Comparing two quantities
- Shifting, rotating bits in a quantity
- Testing, comparing, and converting bits

Types of Processor Operations

Data Movement Operations

- Moving data from memory to the CPU
- Moving data from memory to memory
- Input and output

Types of Processor Operations

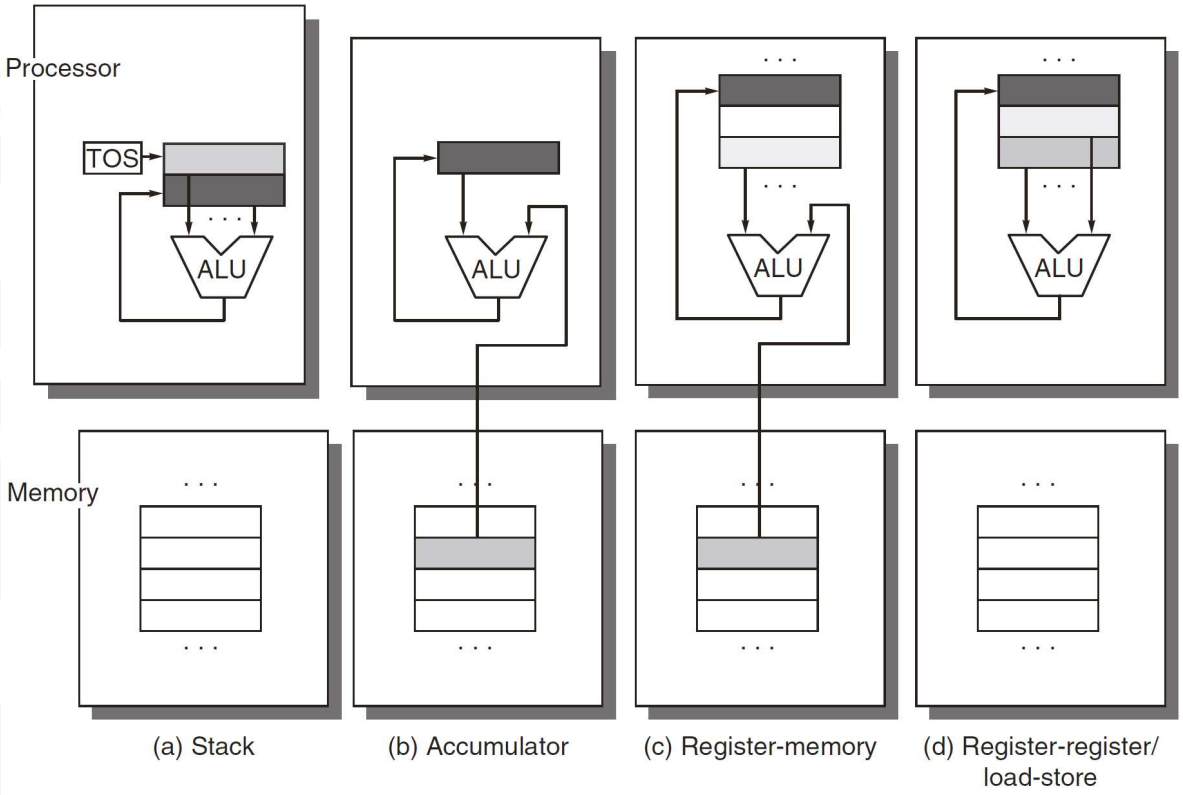
Program Control Operations

- Starting a program
- Halting a program
- Skipping to other instructions
- Testing data to decide whether to skip over some instructions

Instruction Set Architecture (ISA)

- **Instruction vs Program vs Software**
- **Opcode, Operand**
- **Classification of ISA**
  - Stack architecture
  - Accumulator architecture
  - Register-Memory architecture
  - Register-Register/Load-Store architecture

Instruction Set Architecture (ISA)



Operand locations for four instruction set architecture classes.

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

The code sequence for C = A + B for four classes of instruction sets.

Instruction Set Architecture (ISA)

A=D*(B+C)-E

Stack Machine

- Push D
- Push B
- Push C
- Add
- Mul
- Push E
- Sub
- Pop A

Accumulator Machine

- Load B
- Add C
- Mul D
- Sub E
- Store A

Load-Store Machine

- Load R1, D
- Load R2, B
- Load R3, C
- Add R4, R2, R3
- Mul R5, R1, R4
- Load R6, E
- Sub R7, R5, R6
- Store R7, A

Addressing Modes

The way by which an operand is specified in an instruction

Addressing mode	Example instruction	Meaning	When used
Register	`Add R4,R3`	`Regs[R4] ← Regs[R4] + Regs[R3]`	When a value is in a register.
Immediate	`Add R4,#3`	`Regs[R4] ← Regs[R4] + 3`	For constants.
Displacement	`Add R4,100(R1)`	`Regs[R4] ← Regs[R4] + Mem[100 + Regs[R1]]`	Accessing local variables (+ simulates register indirect, direct addressing modes).
Register indirect	`Add R4,(R1)`	`Regs[R4] ← Regs[R4] + Mem[Regs[R1]]`	Accessing using a pointer or a computed address.
Indexed	`Add R3,(R1 + R2)`	`Regs[R3] ← Regs[R3] + Mem[Regs[R1] + Regs[R2]]`	Sometimes useful in array addressing: R1 = base of array; R2 = index amount.
Direct or absolute	`Add R1,(1001)`	`Regs[R1] ← Regs[R1] + Mem[1001]`	Sometimes useful for accessing static data; address constant may need to be large.
Memory indirect	`Add R1,@(R3)`	`Regs[R1] ← Regs[R1] + Mem[Mem[Regs[R3]]]`	If R3 is the address of a pointer p, then mode yields $*p$.
Autoincrement	`Add R1,(R2)+`	`Regs[R1] ← Regs[R1] + Mem[Regs[R2]]` `Regs[R2] ← Regs[R2] + d`	Useful for stepping through arrays within a loop. R2 points to start of array; each reference increments R2 by size of an element, d.
Autodecrement	`Add R1, –(R2)`	`Regs[R2] ← Regs[R2] – d` `Regs[R1] ← Regs[R1] + Mem[Regs[R2]]`	Same use as autoincrement. Autodecrement/-increment can also act as push/pop to implement a stack.
Scaled	`Add R1,100(R2)[R3]`	`Regs[R1] ← Regs[R1] + Mem[100 + Regs[R2] + Regs[R3] * d]`	Used to index arrays. May be applied to any indexed addressing mode in some computers.

Selection of addressing modes with examples, meaning, and usage.

Summary

Suggested Readings

Computer Architecture: A Quantitative Approach (5th Edition) by John Hennessy and David A Patterson:

- pp. 2 – 16: Sections 1.1 – 1.3
- pp. A2 – A16: Sections A.1 – A.5 Instruction Set Principles