Chapter 05: Basic Processing Units … Control Unit Design Organization

Lesson 12:

Sequencing of Control Signals

Objective

- Understand the four sequences of control signals─ (a) Fetch the instruction from memory address I pointed by PC multiple bus organisation

(b) Read the operand from memory address x pointed by ri

(c) Perform Arithmetic operations on the operands

(d) Write back the TEMP to the memory address pointed by ri

Four sequences of control signals

a. Fetch the instruction from memory address I pointed by PC

1. *Step j:* PC → MAR
2. *Step j + 1:* PC ← PC + 4 for 32-bits memory word alignment
3. *Step j + 2:* Activate signal ALE for one cycle
4. *Step j + 3:* Activate signal MEMRD
5. *Step j + 4:* M[I] → MDR
6. *Step j + 5:* Deactivate signal MEMRD
7. *Step j + 6:* MDR → IR

b. Read the operand from memory address *x* pointed by *ri*

1. Step j1: ri → MAR
2. Step j1 + 1: No action as PC already ready for the next instruction
3. Step j1 + 2: Activate signal ALE for one cycle
4. Step j1 + 3: Activate signal MEMRD
5. Step j1 + 4: M[x] → MDR
6. Step j1 + 5: Deactivate signal MEMRD
7. Step j1 + 6: MDR transfers through internal bus— MDR → TEMP (a temporary register for Operand X)

c. Perform Arithmetic operations on the operands

1. Step i: TEMP → X.
2. Step i + 1: X → ALU
3. Step i + 2: rj → Y
4. Step i + 3: Y → ALU
5. Step i + 4: Selects through a gates ϕ_i an operation for SUB at the ID for instruction received at IR during **a** steps j to j + 6

c. Perform Arithmetic operations on the operands

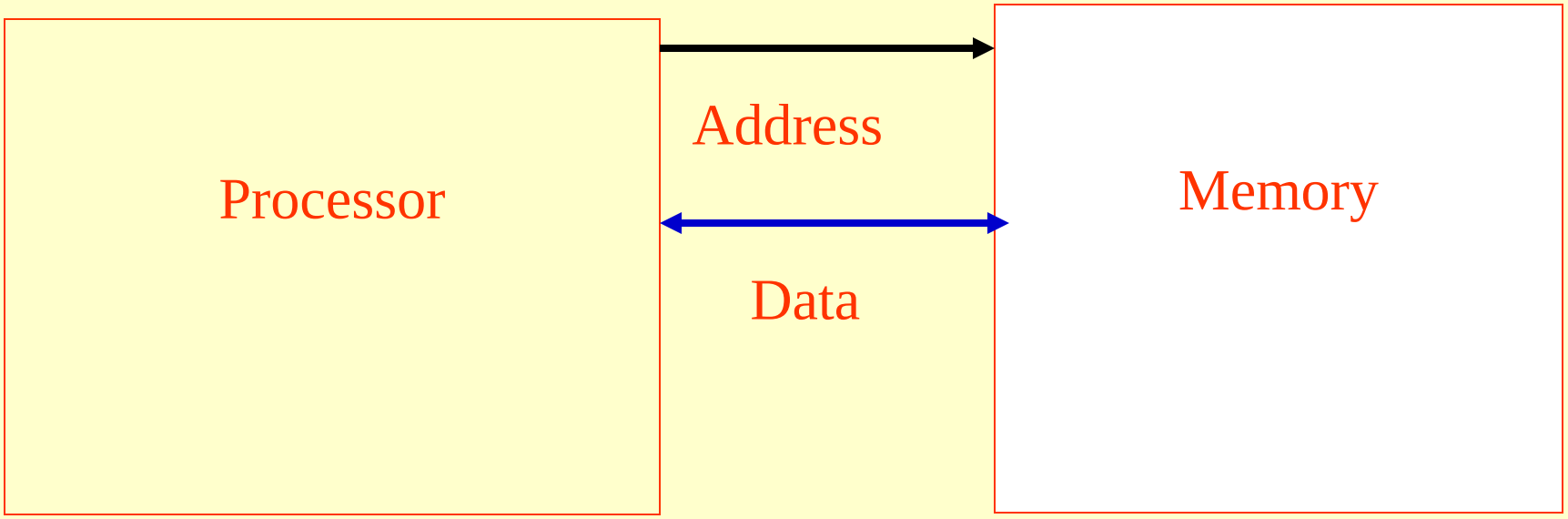
6. Step i + 5: (Z) ← ALU.
7. Step i + 6: Transfers status flags generated; borrow and overflow to status register─ (Status Register) ← ALU
8. Step i + 7: TEMP ← Z

d. Write back the TEMP to the memory address pointed by r_i

1. Step k: Address at ri →MAR.
2. Step k + 1: Activate ALE
3. Step k + 2: Activate signal MEMWR
4. Step k + 3: TEMP → MDR
5. Step k + 4: MDR → data to address at ri
6. Step k + 5: Deactivate signal MEMWR

Six steps in the instruction processing

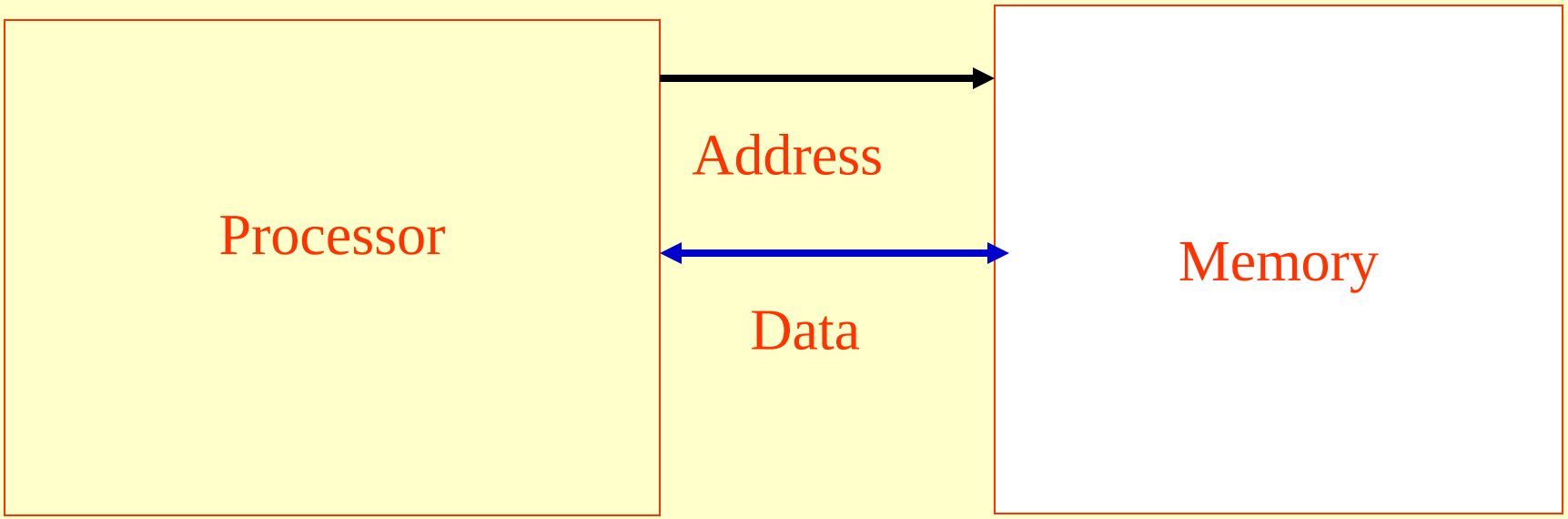
Instruction Fetch



Step 1) Processor requests instruction from memory using address in Program Counter PC register

Step 2) Memory using data bus returns the instruction to instruction register IR register

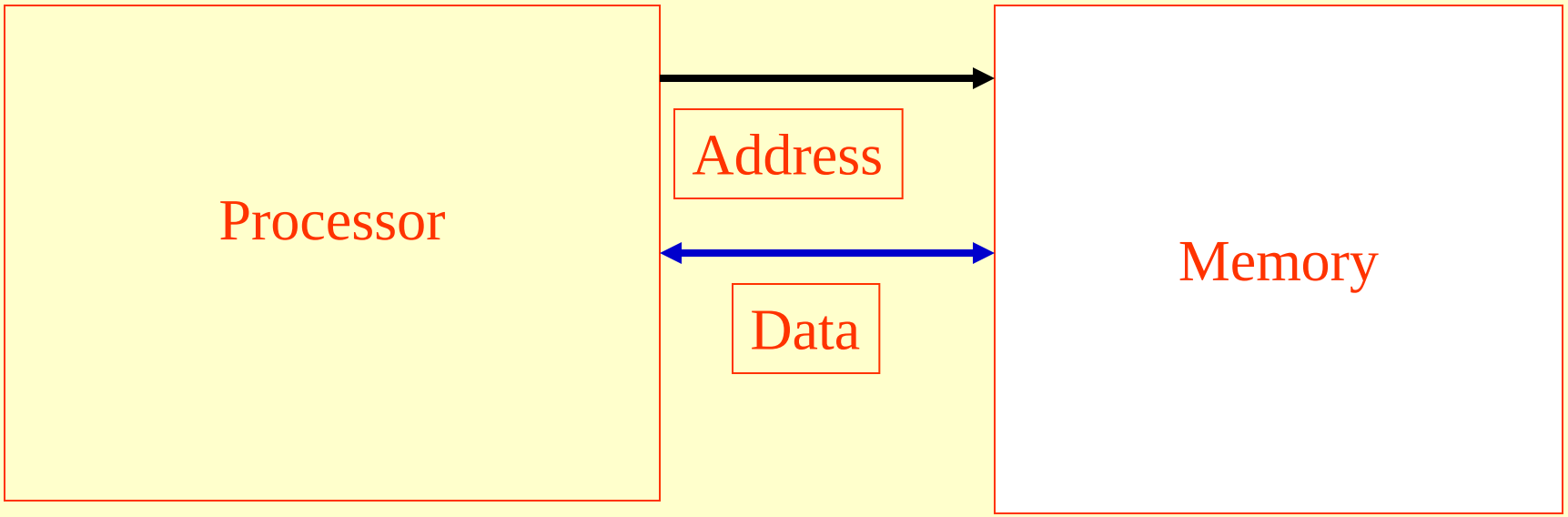
Instruction decoding and execution



Step 3) Processor decodes instruction and places at instruction decoder ID register

Step 4) Processor executes instruction at execution unit

Result Write back and PC update for the next



Step 5) Result of instruction written back for register or memory

Step 6) PC updates for next instruction

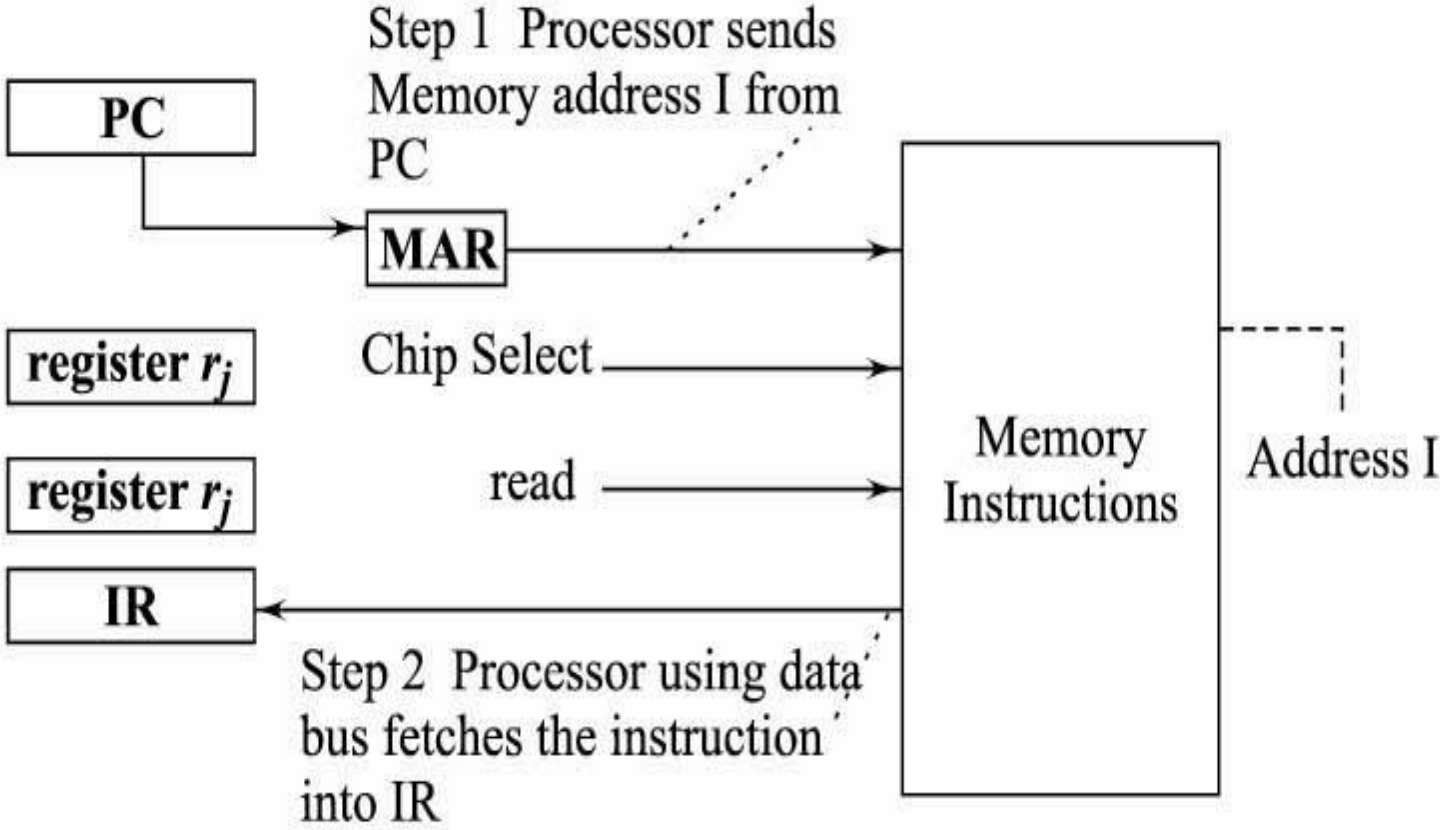
Instruction Fetch and Transfer to Instruction Register

- A step before execution of an instruction is to fetch the instruction into instruction register– directly from memory or through a cache

Instruction Fetch and Instruction Register Transfer

- Let us represent the instruction at a memory address by M[**I**], where **I** is the address of the instruction **I**
- Fetch process representation is then as follows: $I_{addr} \leftarrow PC$, Data_Bus ← M[I] and IR ← Data_Bus

An Instruction Fetch into Instruction Register (IR)



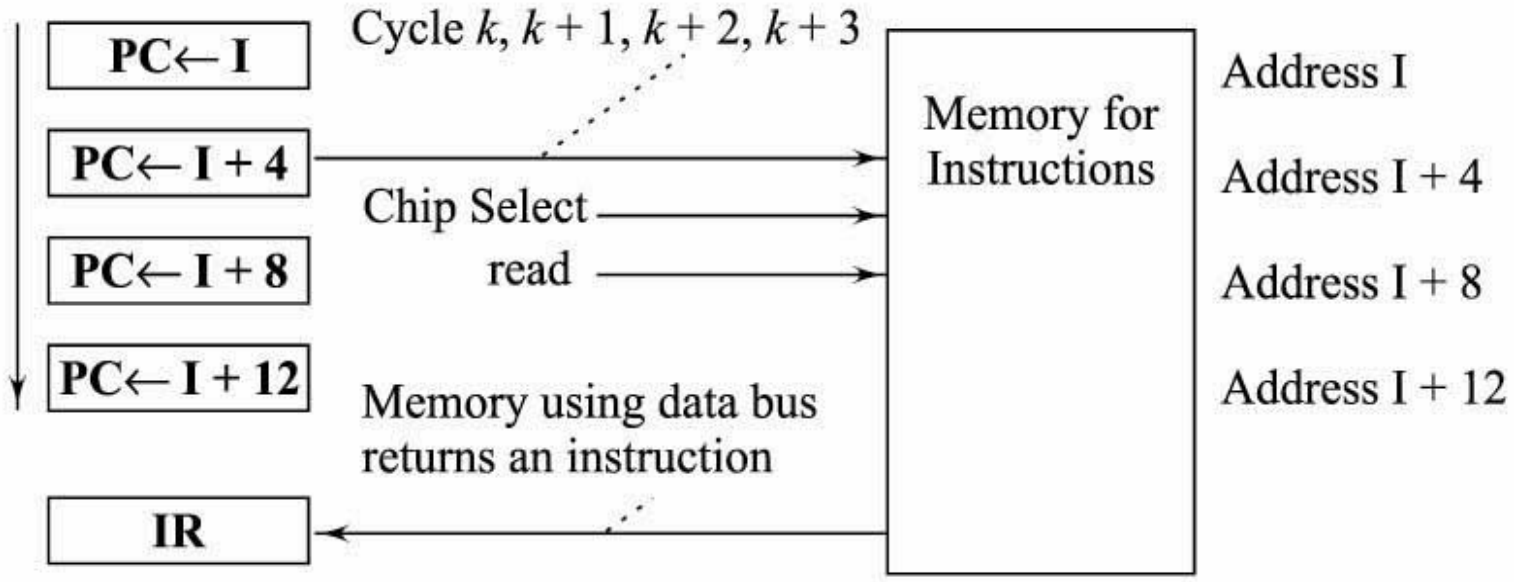
Program Counter Register Increment

- A step before fetch of an instruction– transfer the PC after required increment to fetch the instruction– directly from memory or through a cache

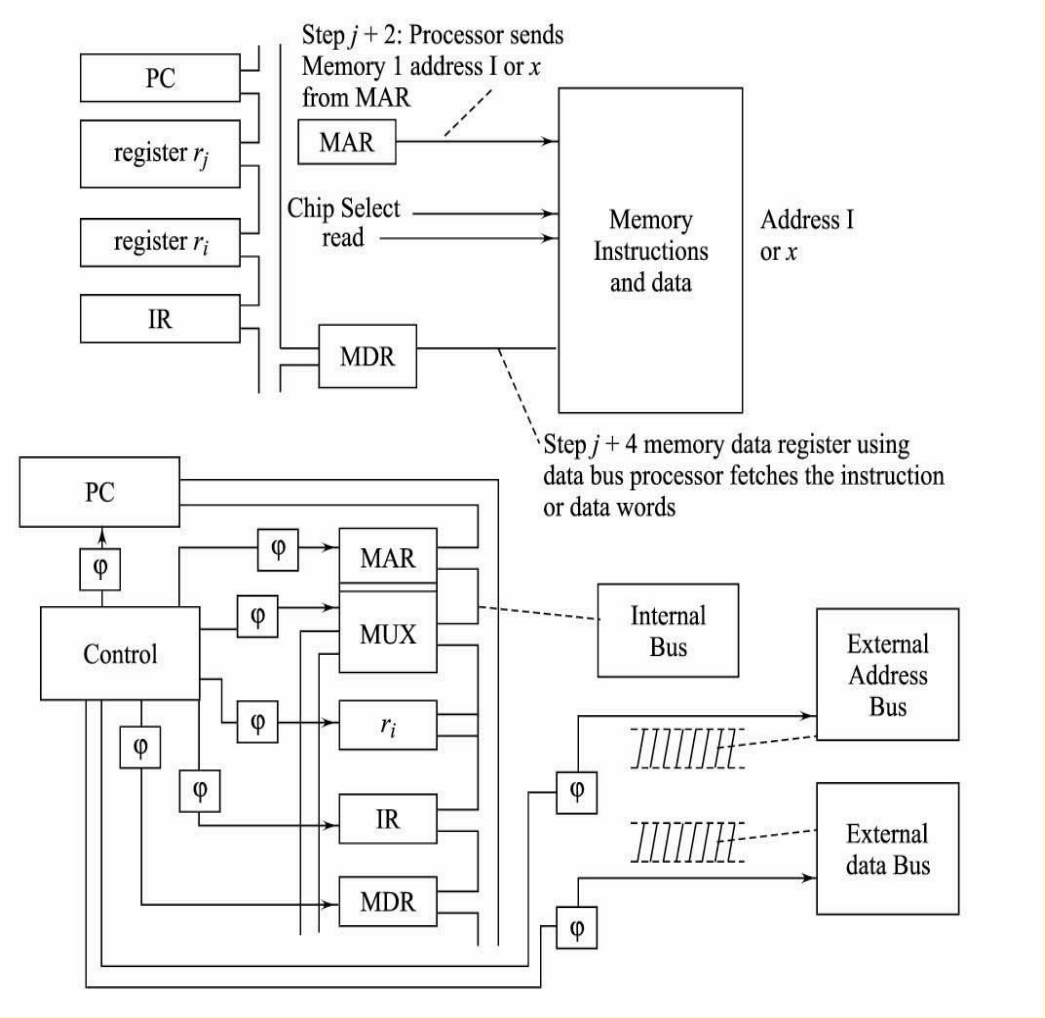
Program Counter Register Increment

- Assume that memory stores *m* byte words at addresses in multiple of 4
- Let us represent the sequences of microoperations in fetch process for a new cycle after the program counter increment as follows: **I** ← I + *m*, $\mathbf{I}_{addr}$ ← PC, Data_Bus ← M[I] and IR ← Data_Bus

Sequence of Program counter during 4 instruction cycles



Fetch a Word from Memory and Transfer to IR or GPR or other Word Storing Unit



Sequences of control signals for implementing six steps in the instruction processing

Step in fetching instruction

- The logic results at register ID initiates a sequence of control actions. Each control signal selects an action through a gate *j* at each step. The sequencing of steps is as follows:

  1. *Step j*: PC transfers to MAR through internal bus. (MAR means a memory address register for sending address bits to the address bus). PC →MAR

Step in fetching instruction

2. *Step j* + 1: If this is also an instruction for the last byte fetch operation, then increment PC to make it ready for the next instruction, which will execute after this instruction completes

- For instruction last byte fetch case, PC ← PC + 4 for 32-bit memory-word alignments

Step in fetching instruction

3. *Step j* + 2: Activate signal address latch enable (ALE) for one cycle and this enables MAR to bus so that MAR bits carry the address bus signals to memory through a latch

- A chip select then activates the memory and ALE deactivates after the cycle. [ALE is needed when address and data bus bits multiplex. After the cycle, the same signals will carry data signals

Step in fetching instruction

4. *Step j* + 3: Activate signal for memory for a memory read (MEMRD) operation
5. *Step j* + 4: Memory transfers opcode or operand to MDR through data bus. (MDR means memory data register for input-output to data bus.) M[I] or M(x)® MDR depending on whether I or x was put in MAR in a step

Step in fetching instruction

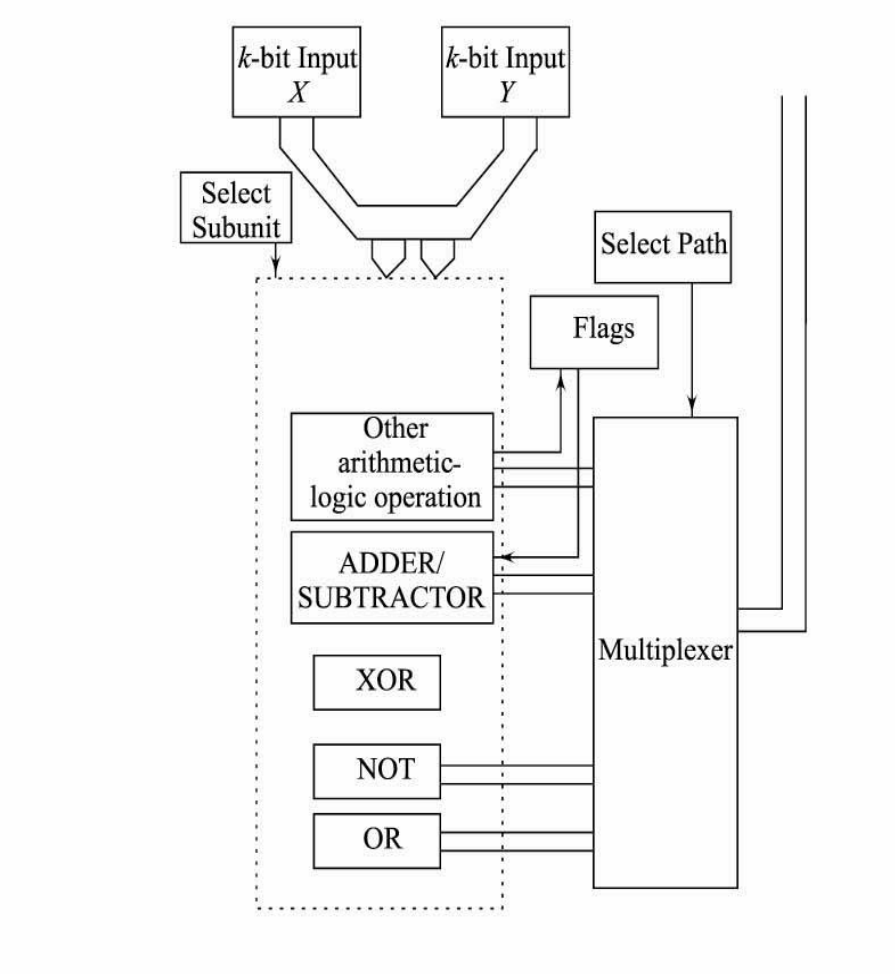
6. *Step j* + 5: Deactivate signal MEMRD.
7. *Step j* + 6: MDR transfers through internal bus. MDR ® IR, GPR, or another word storing-unit through bus or a MUX

- Repeat steps if instruction's operands fetch is complete

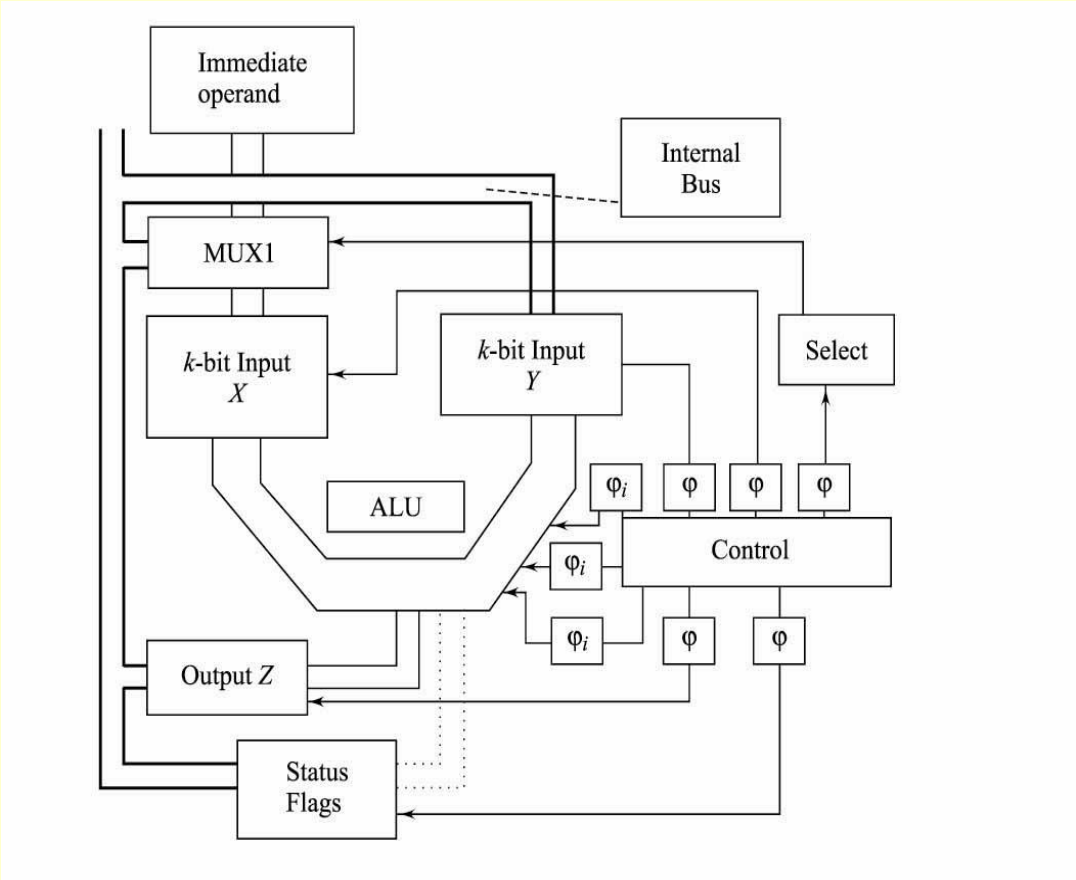
Step in fetching instruction

- An opcode or operands read operation for an instruction completes in seven or more steps
- The targeted word transfers to IR or GPR *ri* or another word storing-unit
- For example one for an immediate operand) and the PC is ready to fetch the next instruction in case that happens to be a cycle for the fetch instruction

Performing an Arithmetic or Logical Operation



ALU design as data path with a control unit for arithmetic or logic operations for destination operand ← Operation (X and Y operands) at ALU



Sequences to perform an arithmetic or logic operation

- After receiving the instruction at IR— decoded by decoding logic
- Then the logic results at register instruction decoder (ID) initiate control actions
- Each control signal selects an action through a gate ϕ at each step

Sequences to perform an arithmetic or logic operation

1. *Step i*: Transfers a k-bit input source operand through the bus or immediate operand to X. Input operand through a MUX → X

   [MUX means a multiplexer to select one among several channels at inputs as per the select subunit signal]

2. *Step i* + 1: *X transfers k-bit input X to* ALU— *X* → ALU

Sequences to perform an arithmetic or logic operation

3. *Step i* + 2: Another input operand transfers k-bits to *Y* (through bus). Input operand $\rightarrow$ *Y*
4. *Step i* + 3: Transfer *Y* to ALU— *Y* $\rightarrow$ ALU
5. *Step i* + 4: Selects through one of the gates ϕ_i an operation as per the arithmetic or logic instruction, which was received at the IR

Sequences to perform an ALUoperation

- 6. *Step i* + 5: Transfers a *k*-bit output *Z* from ALU. *Z* ← ALU.
- 7. *Step i* + 6: Transfers status flags generated, for example, carry or overflow to status register— Status Register ← ALU
- 8. *Step i* + 7: Transfers from *Z* the result to destination operand through bus— (Bus) ← *Z*

Sequences to perform an ALU operation

- ALU instruction [$Z \leftarrow$ operation (X and Y operands)] completes in eight steps

Example of add Operation

- **Add**: Add microoperation is an operation in which two inputs X and Y are taken by adder unit in the ALU and an output Z is given when adder gate select ϕ_{add} activates

Add Operation

- 1. *Step i*: Transfers a k-bit input source operand through the bus or immediate operand to X. Input operand through a MUX→ X [MUX means a multiplexer to select one among several channels at inputs as per the select subunit signal].

  2. *Step i* + 1: *X transfers k-bit input X to* ALU — *X* → ALU-input

Add Operation

3. *Step i* + 2: Another input operand transfers k-bits to *Y* (through bus)─ Input operand →*Y*
4. *Step i* + 3: Transfer *Y* to ALU. *Y* ← ALU-input

Add Operation

5. *Step i* + 4: Selects add gate ϕ_{add} when ADD instruction, which was received at the IR and decoded at the instruction decoder logic —[ϕ_{add} is control input to ALU for ADD]. ϕ_{add} : ALU-output (0-k–1) ← X + Y

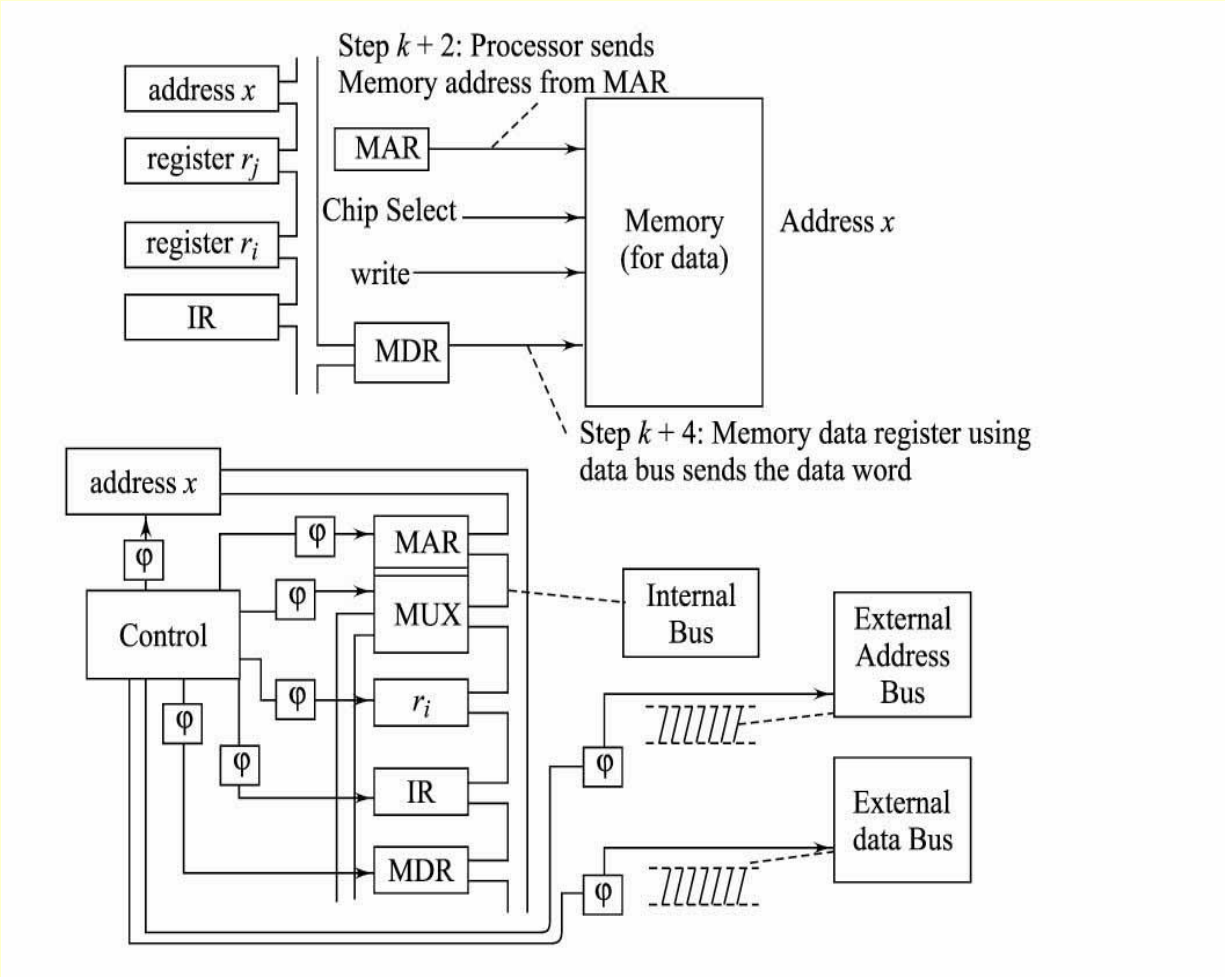
Add Operation

6. *Step i* + 5: Transfers a *k*-bit output *Z* from ALU— *Z* ← ALU-output (0-k–1)
7. *Step i* + 6: Transfers status flags generated, for example, carry or overflow to status register— Status Register ← ALU (status)
8. *Step i* + 7: Transfers from *Z* the result to destination operand through bus— (Bus) ← *Z*

Add Operation

- ALU instruction [$Z \leftarrow$ ADD (X and Y operands)] completes in eight steps

Storing a Word to Memory, Memory Store Using Data Path with a Control Unit



Storing a Word in Memory

- After receiving the instruction at IR, it is decoded by decoding logic
- The logic results at register ID then initiate control actions
- Each control signal selects an action through a gate ϕ at each step

Storing a Word in Memory

1. *Step k:* An address x generated for example, by indirect register address, transfers to MAR through the internal bus. [MAR means a memory address register for sending address bits to the address bus

- The address $x \rightarrow$ MAR

Storing a Word in Memory

2. *Step k + 1:* Activate signal for address latch enable (ALE) for a clock cycle and to enable MAR to bus so that MAR bits carry the address bus signals to memory through the latch

- The chip select then activates at memory and ALE deactivates after one cycle
- [The ALE is needed when address and data bus bits multiplex. The required chip select is from an external decoder that selects the targeted data memory chip among several ones]

Storing a Word in Memory

3. *Step k + 2:* Activate signal for memory for a memory write (MEMWR) operation.
4. *Step k + 3:* Transfer output operand data for memory to MDR through internal bus. (MDR means memory data register for input-output to data bus.) Output operand →MDR
5. *Step k + 4:* MDR transfers. MDR → M[*x*]
6. *Step k + 5:* Deactivate signal MEMWR

Storing a Word in Memory

- Output operand data write operation completes in six steps and the targeted word transfers to address *x*

Summary

We learnt

- Sequence of control signals for fetching an instruction, arithmetic operation and storing of the word

End of Lesson 12 on

Sequencing of Control Signals