Module 3-
Processor Organization and Architecture

-Neha Surti

Flip Flop

- A circuit that has two stable states which are used to store binary data that can be changed by applying varying inputs.
- Fundamental building blocks of the digital system
- It is the basic storage element in sequential logic circuit.
- Types:
    1. SR Flip-Flop
    2. D Flip-Flop
    3. JK Flip-Flop
    4. T Flip-Flop

Truth Table, Characteristic Table and Excitation Table of Flip Flop

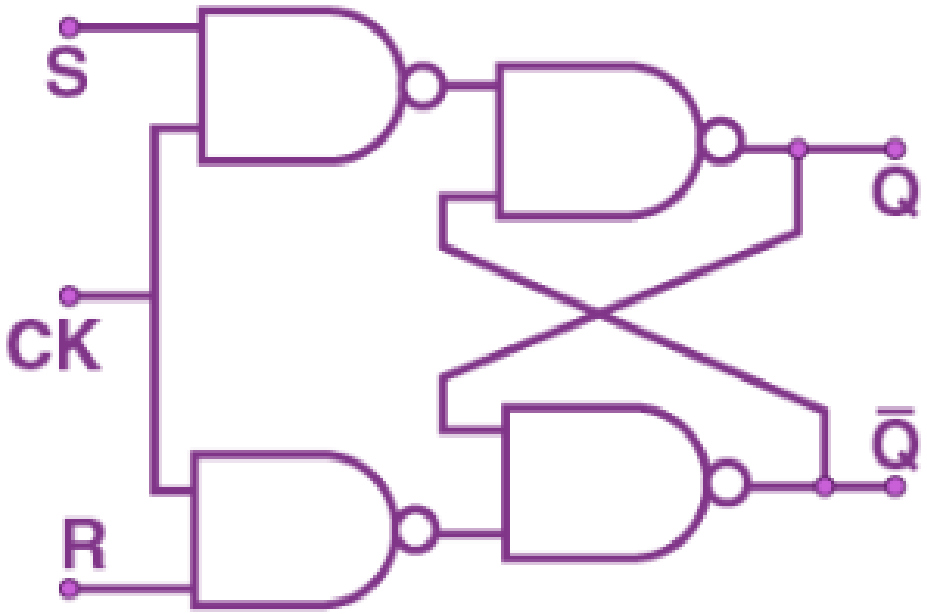
- Truth Table
  - table in between input & output.

- Characteristic Table
  - the next state of flip-flop in terms of flip-flop input and current state

- Excitation Table
  - the flip-flop input variable as function of the current state and next state

SR Flip Flop

- The flip-flop circuit which has a set input (S) and a reset input (R).
- In this system, when you Set S as active, the output Q would be high, and Q′ would be low



Inputs		Output
S	**R**	**Q_{n+1}**
0	0	Qn
1	0	1
0	1	0
1	1	Invalid

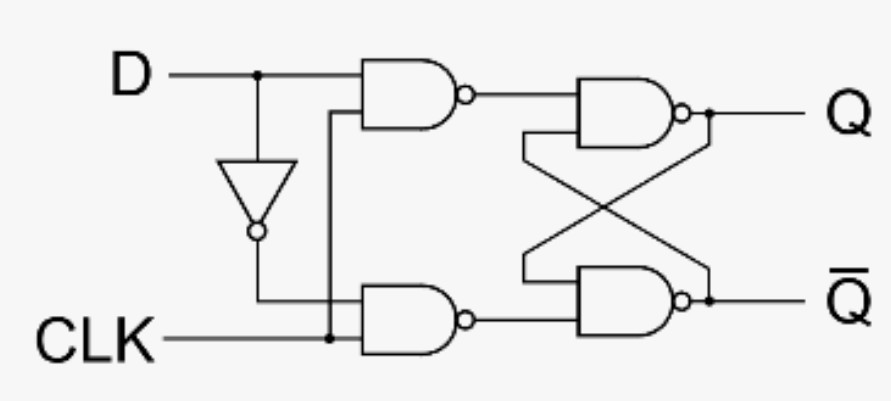
SR Flip Flop

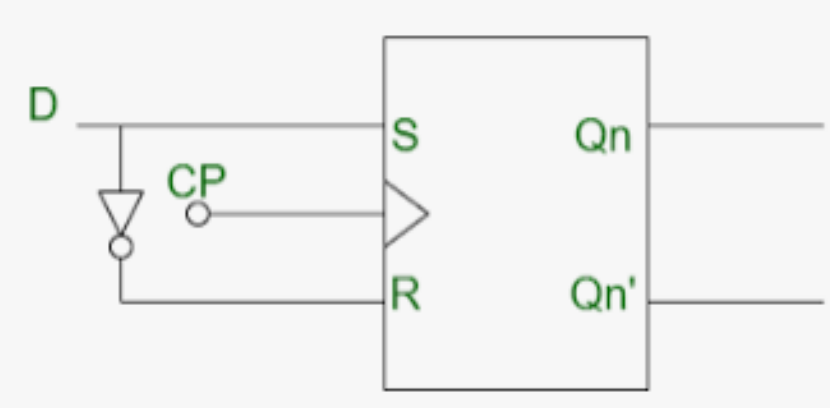
Characteristics table			
Q(n)	S	R	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Invalid
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Invalid

Excitation table			
Q(n)	Q(n+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

D Flip Flop

- It is known as “delay flip flop” or “data flip flop”
- It is used to store single bit of data
- D flip flop consist of a single input D and two outputs (Q and Q’)





D Flip Flop

Clock	D	Q_{n+1}
↓ » 0	X	Q_n
↑ » 1	0	0
↑ » 1	1	1

Truth Table

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

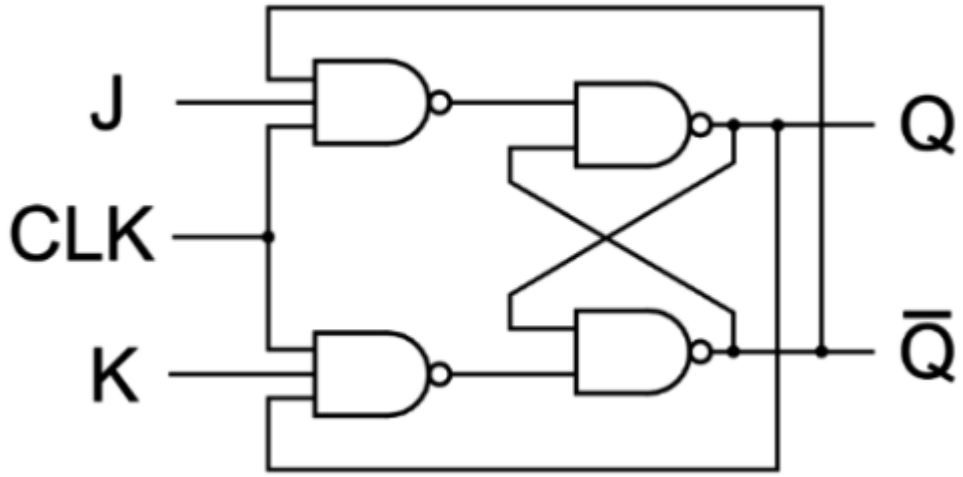
Characteristic Table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

JK Flip Flop

- The JK Flip Flop is a gated SR Flip-Flop with a clock input circuitry that prevents the illegal or invalid output when both inputs S and R are equal to logic level "1."



Clock	J	K	Q_{n+1}	State
0	x	x	Q_n	
1	0	0	Q_n	Hold
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	$\overline{Q}_n$	Toggle

JK Flip Flop

Q(n)	J	K	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

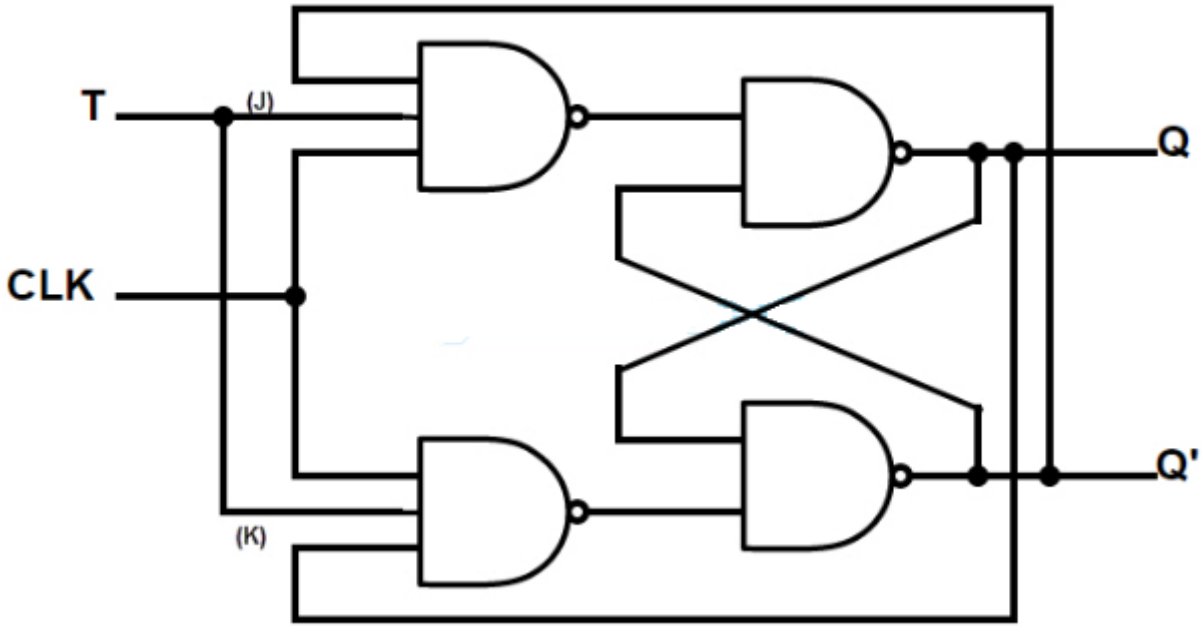
Characteristic Table

Q Output		Inputs	
Present State	Next State	J_n	K_n
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Excitation Table

T Flip Flop

- Toggle Flip Flop-able to toggle its output depending upon on the input.



Inputs		Outputs	
CLK	T	Q_{n+1}	Action
0	X	Qn	No change
1	0	Qn	No change
1	1	$\overline{Q}n$	Toggle

T Flip Flop

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

Register Organization

The registers in the processor perform two roles:

- **User-visible registers:** Enable the machine- or assembly language programmer to minimize main memory references by optimizing use of registers.
  - General purpose
  - Data
  - Address
  - Condition codes

- **Control and status registers:** Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs.

Register Organization

- **Control and status registers:**
  Registers used for the movement of data between the processor and memory:
  - Program counter (PC) Contains the address of an instruction to be fetched
  - Instruction register (IR): Contains the instruction most recently fetched
  - Memory address register (MAR): Contains the address of a location in memory
  - Memory buffer register (MBR): Contains a word of data to be written to memory or the word most recently read

  Many processor designs include a register or set of registers, often known as the **program status word (PSW)**, that contain status information. Common fields or flags include the following:
  - Sign
  - Zero
  - Carry
  - Equal
  - Overflow
  - Interrupt Enable/Disable
  - Supervisor

Instruction Formats

- An instruction format defines the layout of the bits of an instruction, in terms of its constituent fields.
- An instruction format must include an opcode and, implicitly or explicitly, zero or more operands.
- The most common fields are:
  - Operation field specifies the operation to be performed like addition.
  - Address field which contains the location of the operand, i.e., register or memory location.
  - Mode field which specifies how operand is to be founded.
- There are several types of instruction formats, including zero, one, two, and three-address instructions.

Instruction Formats

Zero-address instructions:

- Do not specify any operands or addresses
- Operate on data stored in registers or memory locations implicitly defined by the instruction
- For e.g., Stack is included in the CPU for performing arithmetic and logic instructions with no addresses. The operands are pushed onto the stack from memory and ALU operations are implicitly performed on the top elements of the stack.

One Address Instructions:

- This address is of the first operand.
- The second operand and the result are stored in a CPU register called Accumulator Register (AR)

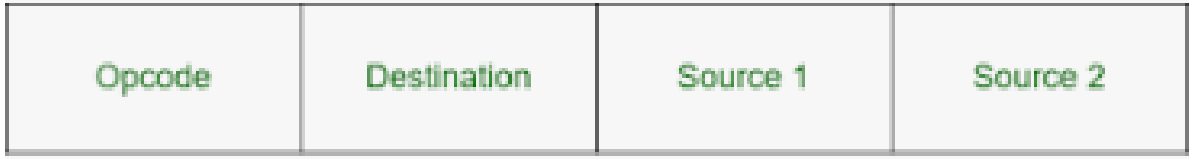
Instruction Formats

Two Address Instructions:

- In this format, two addresses and an operation field is there.
- The result is stored in either of the operand address i.e., either in address of first operand or in the address of second operand.

Three Address Instructions:

- This system contains three address fields (address of operand1, address of operand2 and address where result needs to be put).



Addressing modes

- The address field or fields in a typical instruction format are relatively small.
- To reference a large range of locations in main memory, a variety of addressing techniques has been employed.
- The most common addressing techniques:
  - Immediate
  - Direct
  - Indirect
  - Register
  - Register indirect
  - Displacement
  - Stack

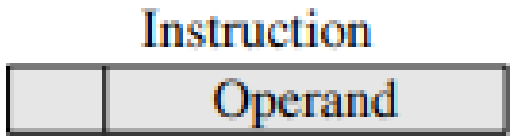
A = contents of an address field in the instruction
R = contents of an address field in the instruction that refers to a register
EA = actual (effective) address of the location containing the referenced operand
(X) = contents of memory location X or register X

Addressing modes

Immediate Addressing :

- The simplest form of addressing in which the operand value is present in the instruction

  Operand = A

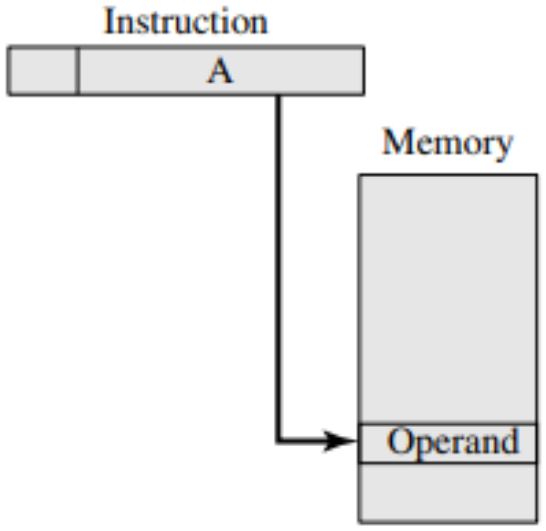


- This mode can be used to define and use constants or set initial values of variables.
- Advantage: no memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle.
- Disadvantage: the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length

Addressing modes

Direct Addressing :

- The address field contains the effective address of the operand:
  EA = A
- It requires only one memory reference and no special calculation.
- Limitation: it provides only a limited address space.

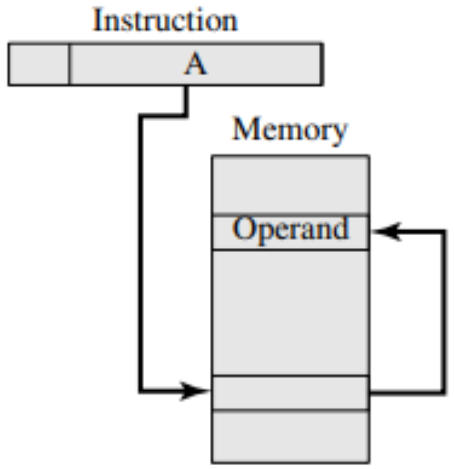


Addressing modes

Indirect Addressing :

- With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range.
- One solution is to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand.

  EA = (A)
- Disadvantage: instruction execution requires two memory references to fetch the operand: one to get its address and a second to get its value.



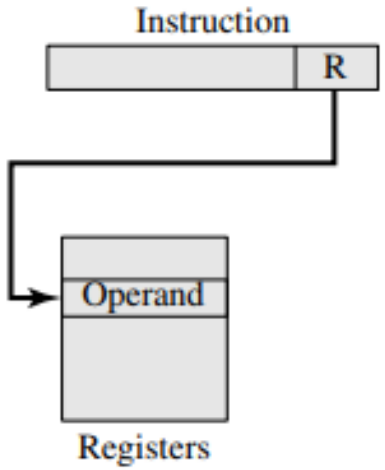
Addressing modes

Register Addressing :

- Register addressing is similar to direct addressing.
- The only difference is that the address field refers to a register rather than a main memory address:

  EA = R
- Advantages: (1) only a small address field is needed in the instruction, and

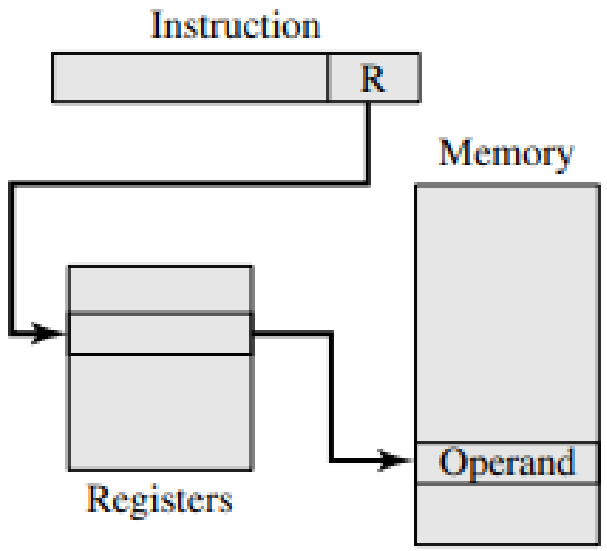
  (2) no time-consuming memory references are required.



Addressing modes

Register Indirect Addressing :

- Register indirect addressing is analogous to indirect addressing.
  EA = (R)
- Advantages and limitations are same as for indirect addressing.
- It uses one less memory reference than indirect addressing.



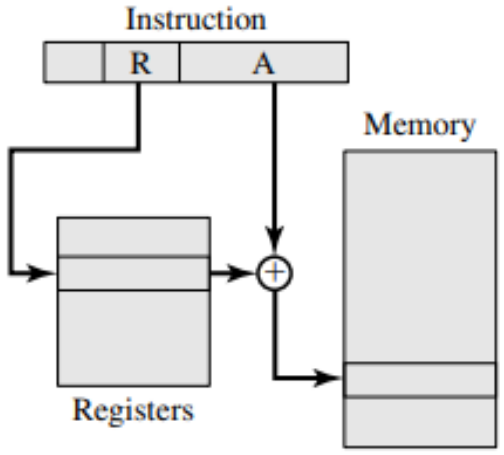
Addressing modes

Displacement Addressing :

- A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing.

  EA = A + (R)

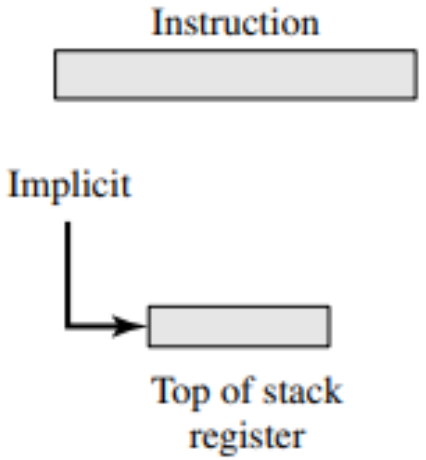
- It requires that the instruction have two address fields, at least one of which is explicit.
- The value contained in one address field (value = A) is used directly. The other address field, or an implicit reference based on opcode, refers to a register whose contents are added to A to produce the effective address.
- Relative addressing
- Base-register addressing
- Indexing



Addressing modes

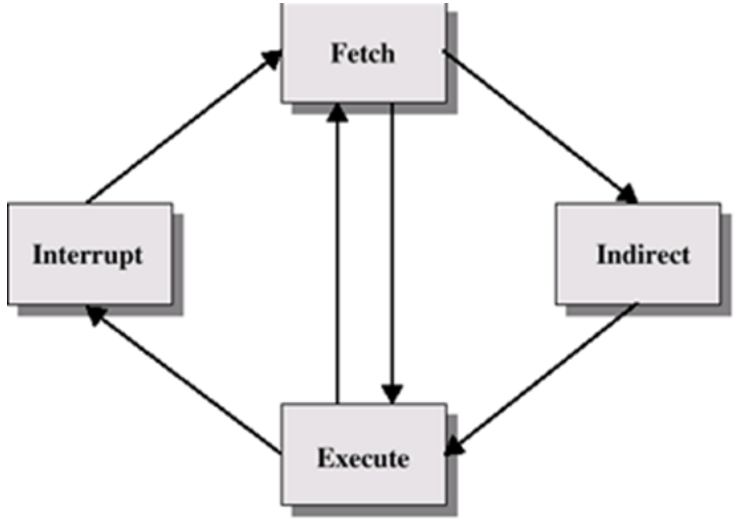
Stack Addressing :

- The stack mode of addressing is a form of implied addressing.
- The machine instructions need not include a memory reference but implicitly operate on the top of the stack.
- Associated with the stack is a pointer whose value is the address of the top of the stack.



Instruction Cycle

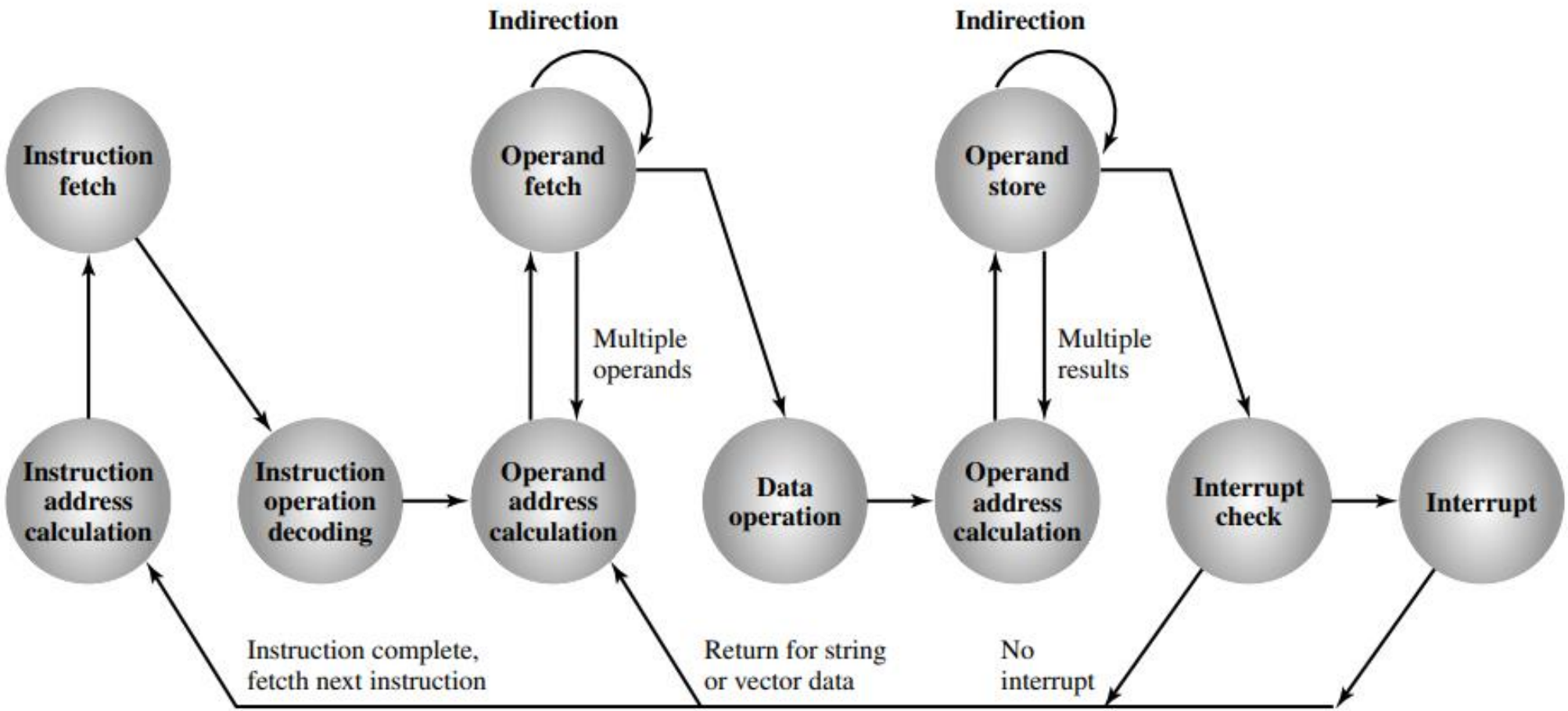
- An instruction cycle includes the following stages (as shown in fig):
  - Fetch: Read the next instruction from memory into the processor.
  - Execute: Interpret the opcode and perform the indicated operation.
  - Interrupt: If interrupts are enabled and an interrupt has occurred, save the current process state and service the interrupt.



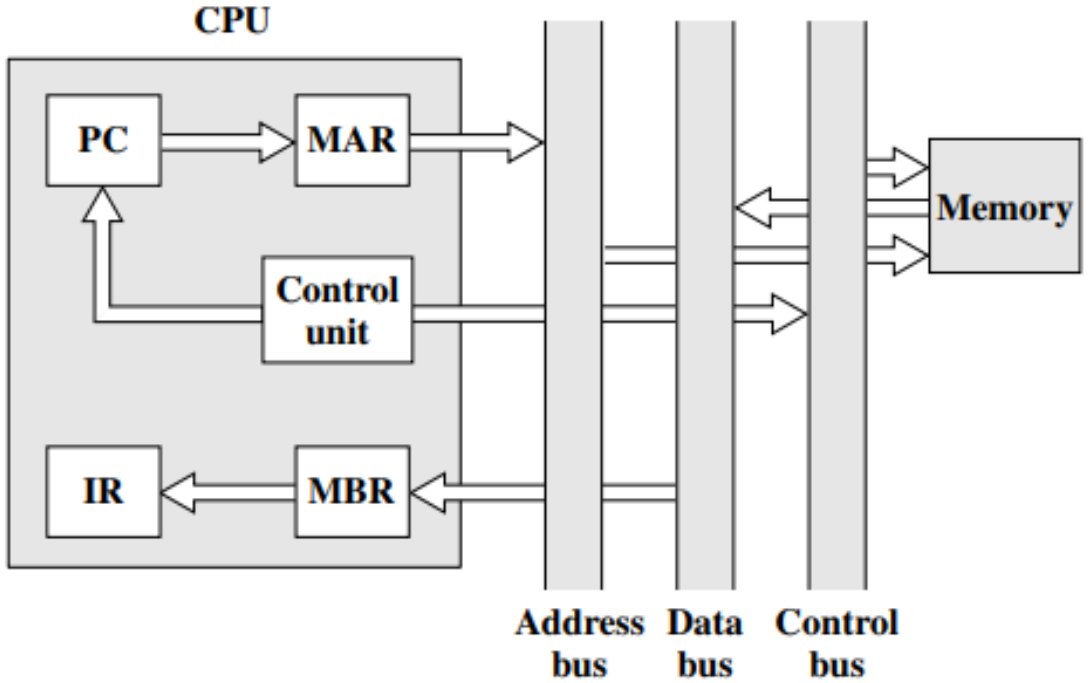
Indirect Cycle

- The execution of an instruction may involve one or more operands in memory, each of which requires a memory access.
- Further, if indirect addressing is used, then additional memory accesses are required.
- After an instruction is fetched, it is examined to determine if any indirect addressing is involved.
- If so, the required operands are fetched using indirect addressing.
- Following execution, an interrupt may be processed before the next instruction fetch.

Instruction Cycle State Diagram



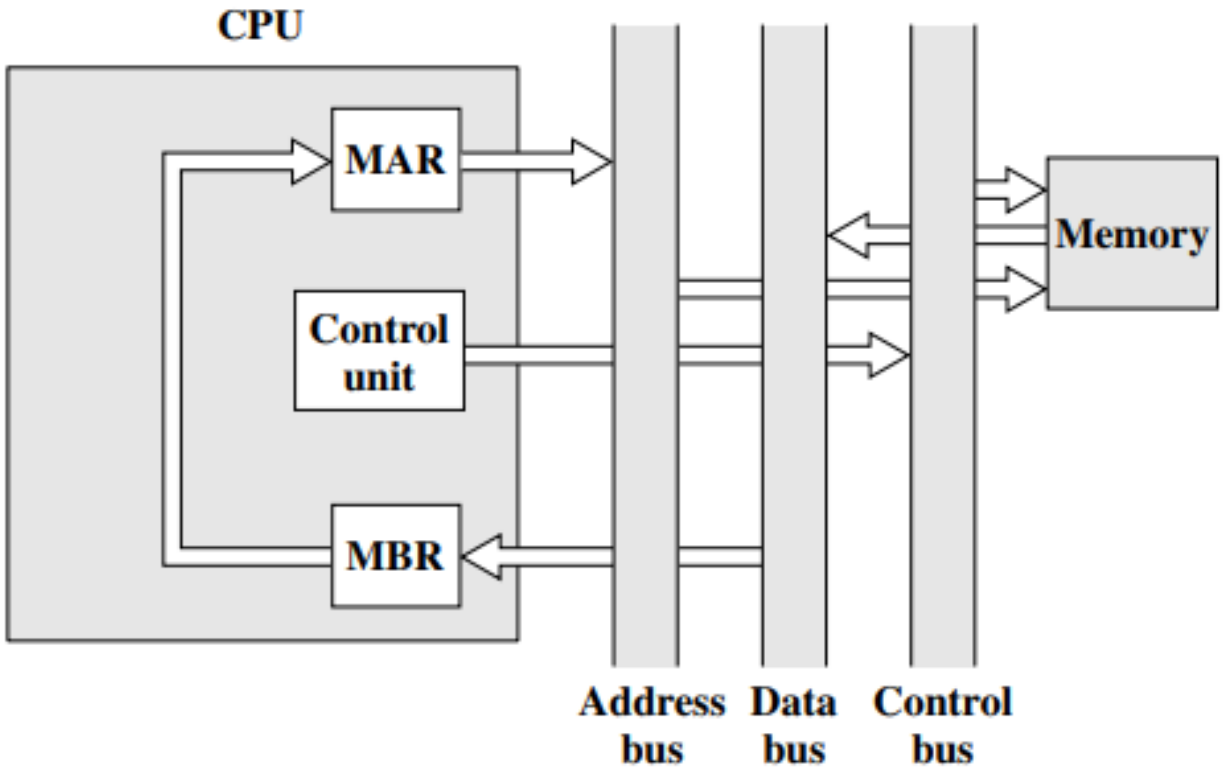
Instruction Cycle



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

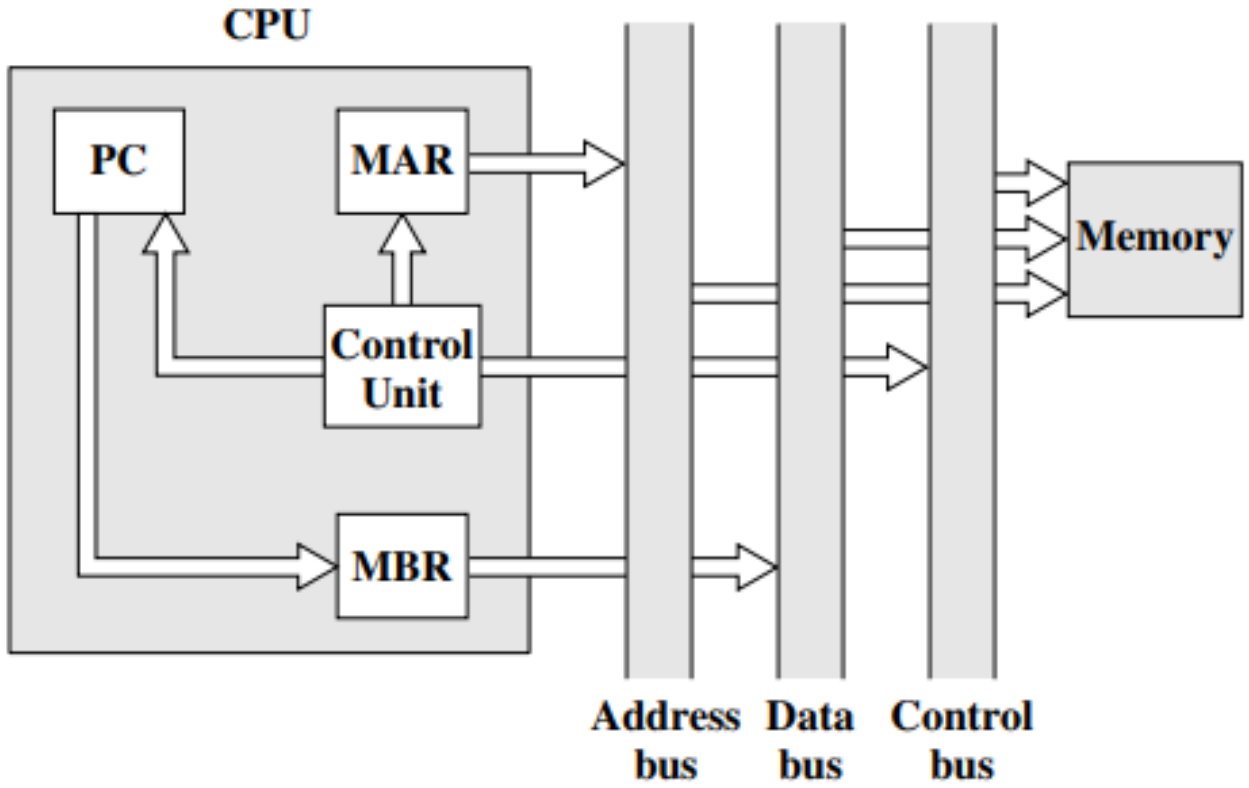
Data Flow, Fetch Cycle

Instruction Cycle



Data Flow, Indirect Cycle

Instruction Cycle



Data Flow, Interrupt Cycle

Thank You