UNIT 6:

Processor Organization

To understand the organⁿ of CPU. Let us consider requiᵗ placed on the CPU. The things that it must to i) Fetch instruction : The CPU reads an instruction from memory

ii) Interpret Instrucⁿs : The instruction is decoded to determine what action is required.

iii) Fetch data : The execution of an instructⁿ may req. reading data from mem. or an i.o. module

iv) Process data : The execution of an instruction may req. performing some logical and arith. operⁿ on data

v) Write data : The result of an execution may req writing data to memory or an IOM

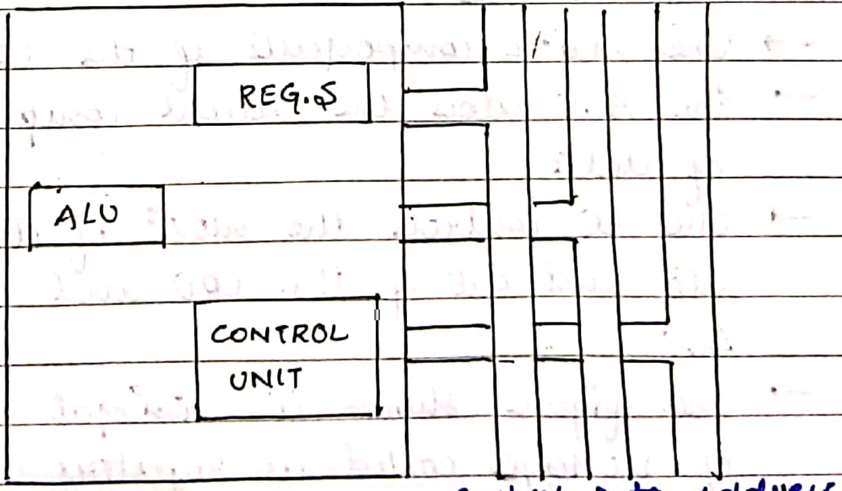


Fig 1 : CPU with system BUS

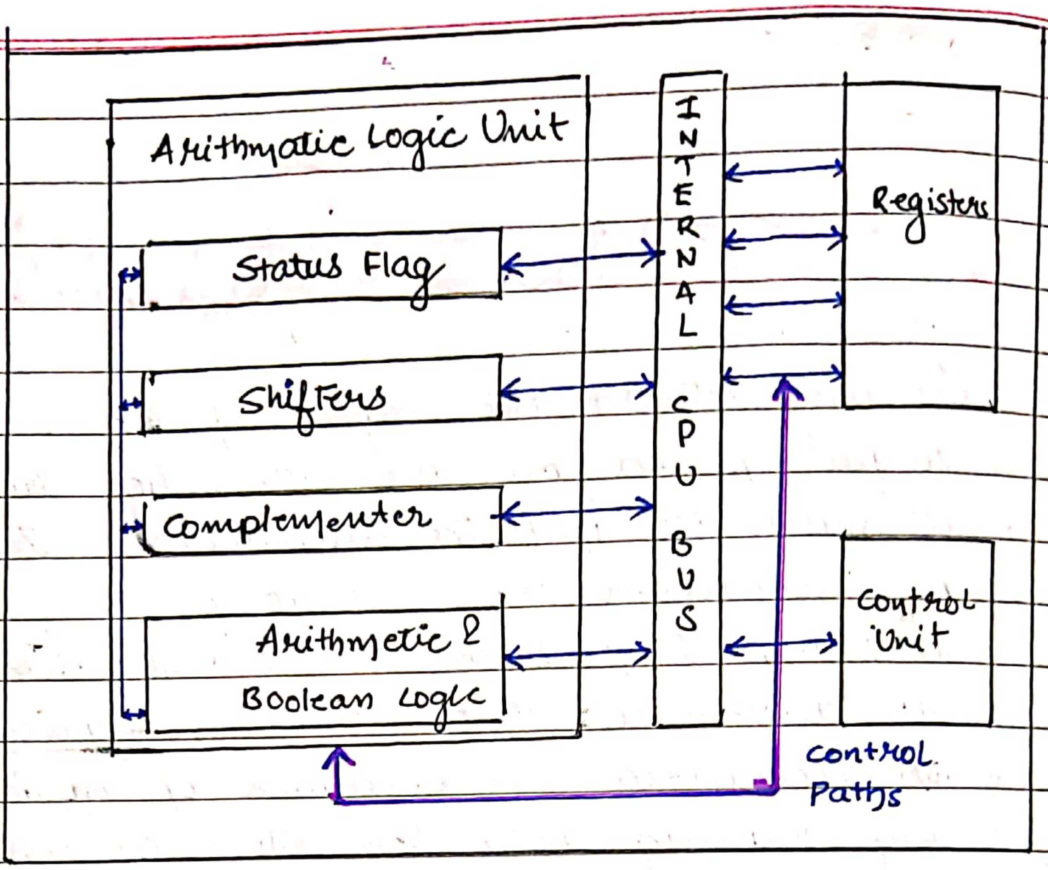


Fig2: Internal Structure of CPU

Fig1:

It shows simplied view of a CPU indicating its connections to the rest of the system via the system bus

→ The major components of the CPU are ALU and CU

→ the ALU does the actual computn or processing of data

→ The CU controls the movt of data and instruction into and out of the CPU and controls operation of ALU

→ The figure shows a minimal internal mem. consisting of storage called as registers

Fig 2:

The fig shows more detailed view of CPU
The data transfer and logic control paths are indicated including an element labeled as internal CPU bus

→ The element is needed to transfer data b/w the various reg. and the ALU

→ There is small collection of elements :

Computer : CPU, IO, Mem.

CPU : CU, ALU, Register connected by data paths

* Register Organization

13 Marks

RO

USED BY PROGRS → User Visible Reg

- General Perpose
- Data
- Address → Segment pointers, Index Regs, Stack Ptrs → RSI, RDI
- Condition Codes
  - ↳ CMP, JUMP...
  - FLAG REG MODIFIES

Control and Status Reg. :

- Program Counter PC
- Instruction Reg
- Mem Add Reg
- Mem Buffer Reg
- PSW (Prog Status Words)
  - → sign
  - → zero
  - → carry
  - → Equal
  - → overflow
  - → Interrupt Enable/Dis
  - → Supervisor

PC - next line

IC - current line which is executing

] Points

The reg. in the CPU performs 2 roles

→ User visible regs.

→ These enable machine / ASM progrs to minimize main mem. references by optimizing use of regs.

→ General Perpose Reg : can be assigned to a varity of func^{n} by the progrs.

→ Any GPR can contain the operand for any Opcode

→ In some case GPR can be used for addressing func^n's like regr indirect, displacement

Data Registers : May be used to hold only data & can not be employed in calcul^n of an operand address

Address Regrs : They are some what GR or they may be devoted to a perticular addressing mode

Eg : Include the following

→ Segment Ptr : In a machine with segmented addressing, a seg reg holds the address of the base of the segments

→ Index Reg : These are used for Index Addressing Mode

→ Stack Pointer : If there is a user visible stack addressing then typically the stack in the mem & there is a decicated reg that points to the top of the stack.

→ Condition Code : Also reffered as flags :

CC are bits set by the CPU hardware as tt result of the oper^n's

Eg : An Arithmatic oper^n may perform a +ve , -ve , zero or overflow result . In add^n to the result itself being stored in a reg or mem. a cond^n C is also set

IN CMP it just change flag does not store result

ADD it changes flag and store result also.

→ Control and Status Reg:

There are a varity of CPU registers that are employed to control the Operⁿ of CPU most of these, on most machine are not visible to the user.

→ Some of them may be visible to machine instrucⁿs executed in a control or OS mode.

→ 4 regs are essential to instruction executⁿ

i> Prog Counter: Contains add of inst. to be fetched

ii> Instruction Reg: Contains the inst most recently fetched

iii> MAR: Mem Add Reg :- Contains the add. of locⁿ in mem.

iv> MBR: Contains a word of data to be written in mem or the word most rec. read.

→ The CPU updates the PC after each instruction fetch so that the PC points to next inst. to be executed

→ A branch or skip inst. will also modifies the content of the PC

→ The fetch inst is loaded into an IR where the opcode & operand specifier are analyzed

→ Data are exchanged with mem using MAR and MBR.

MAR connects dir. to the add. bus
MBR data bus

→ Prog Status Word

All CPU desgn include a reg or a set of regs often k/a PSW that contains status inforⁿ

→ PSW contains cond^n codes + status infor^n, common fields or flags include

- ↝ Sign : Contains sign of the resut of last arithmyatic oper^n
- ↝ zero : Set when resut is zero
- ↝ Carry : Set if an oper^n resulted in a carry (ADD) or borrow (SUB) NO BORROW FLAG

  It is used for multiword arithmyatic inst
- ↝ Equal : Set if logical compare res. is equality
- ↝ Overflow : Used to indicate arithmyatic overflow
- ↝ Interrupt E/D : Used to enable or disable interrupts
- → Supervisor : Indicated whether the CPU is executing in supervisor or usermode

MOVE TO END OF CODE SEGMENT

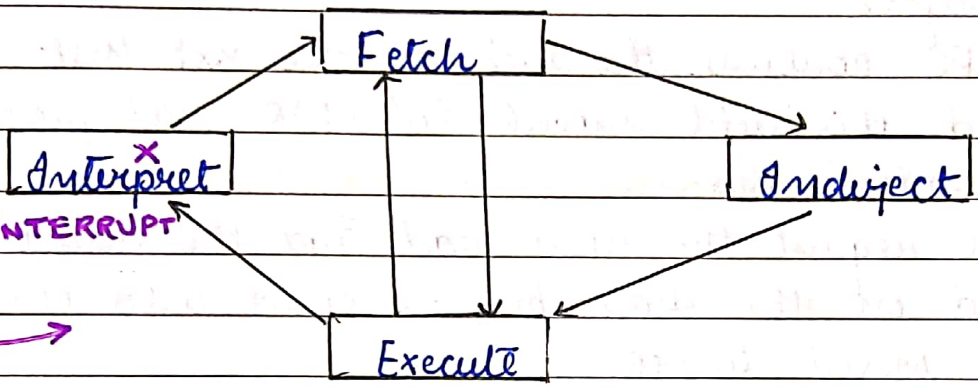
Theory Cont...

* Instruction Cycle

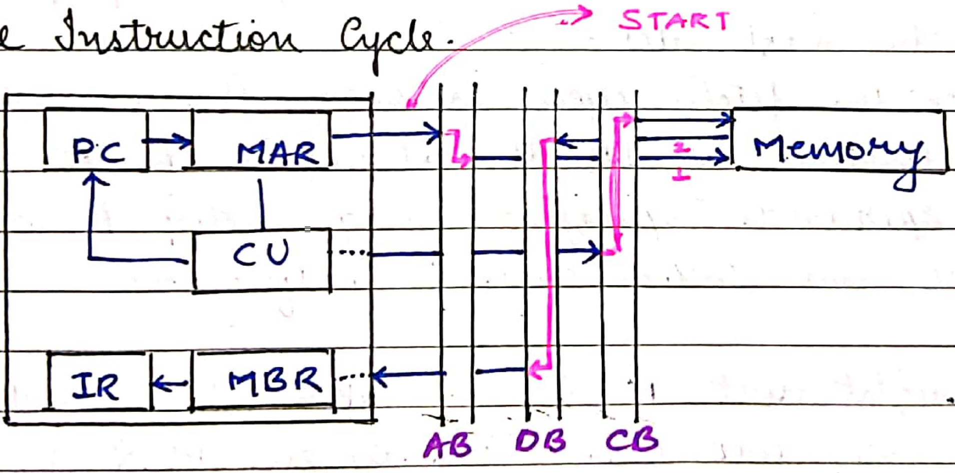
13 MARKS

→ Fetch
→ Execute
→ Interpret x Interrupt.

* Indirect Cycle



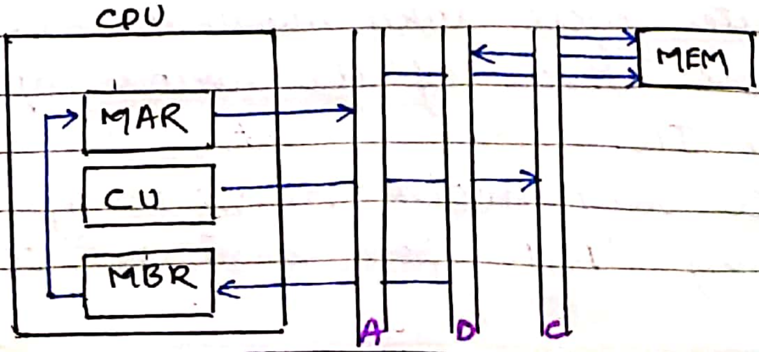
* The Instruction Cycle.



MAR: Memory Address Reg
MBR: Memory Buffer Reg
PC : Prog. Counter
IR: Instruction Reg.

Data flow
fetch cycle

*



Data flow
Indirect Cycle
DETAILED FIG OF
FIG 1

The exact sequence of event during an inst cycle depends on the design of the CPU.

→ Let us assume that a CPU that employees a memory AR, a MBR, PC and Instruction Reg.

→ During the fetch cycle an inst is read from memory figure shows the flow of data during this cycle

→ The PC contains the add of the nxt inst to be fetched this add moved to MAR and placed on the address Bus

→ The CU request the mem read and the result is placed on the data bus & copied into the MBR then moved to IR.

→ Meanwhile the PC is Inted by one preparatory for the next fetch.

→ Once the fetch cycle is over the CU examine the content of IR to determine if it contains an operands specifier using indirect addressing

→ If so an Indir. cycle is performed

3rd dig →The rightmost n bits of MBR which contains the add. ref are tranferred to the MAR +

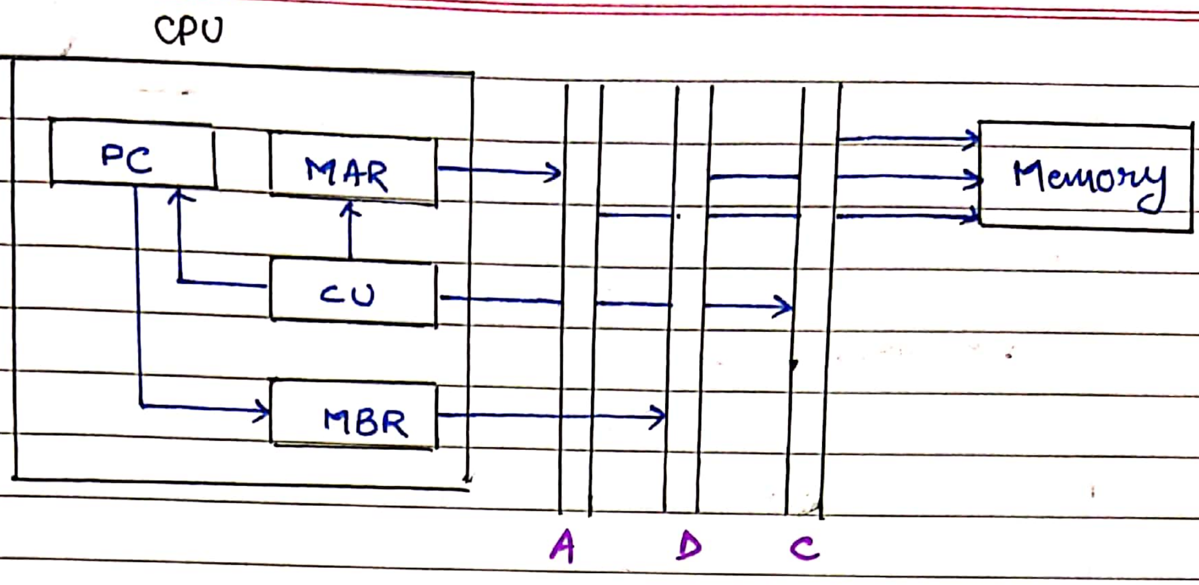
→ Then the control unit req. a mem. read to get the desired add of the operand into the MBR

→ The fetch and indir cycle are simple and predictable

→ The execute cycle takes many forms, the form depends on which of the various machine inst. is in the IR.

→ This cycle may involve transferring data among reg, R/W from mem or IO

*



Data flow Interrupt Cycle.

- The current content of PC must be same so that the CPU can resume normal activity after interrupt.
- Thus the content of PC are transferred to MBR to be written into memory
- The special mem. loc^n ~~X~~ reserved for this ~~task~~ purpose is loaded into MAR from the control Unit
- For Eg. It might be a stack pointer

mov rax, 1
. . .
. . .
syscall
Resume
PC → mov rax
From any task to other (nxt) task where PC is pointing

- The PC is loaded with the add. of the interrupt routine
- As a result nxt inst. cycle will begin by fetching the appropriate inst.