

UNIT II

BASIC PROCESSING UNIT

- ☐ Fundamental concepts
- ☐ Execution of a complete instruction
- ☐ Multiple bus organization
- ☐ Hardwired control
- ☐ Micro programmed control
- ☐ Nano programming.

www.5starnotes.com

Basic fundamental concepts

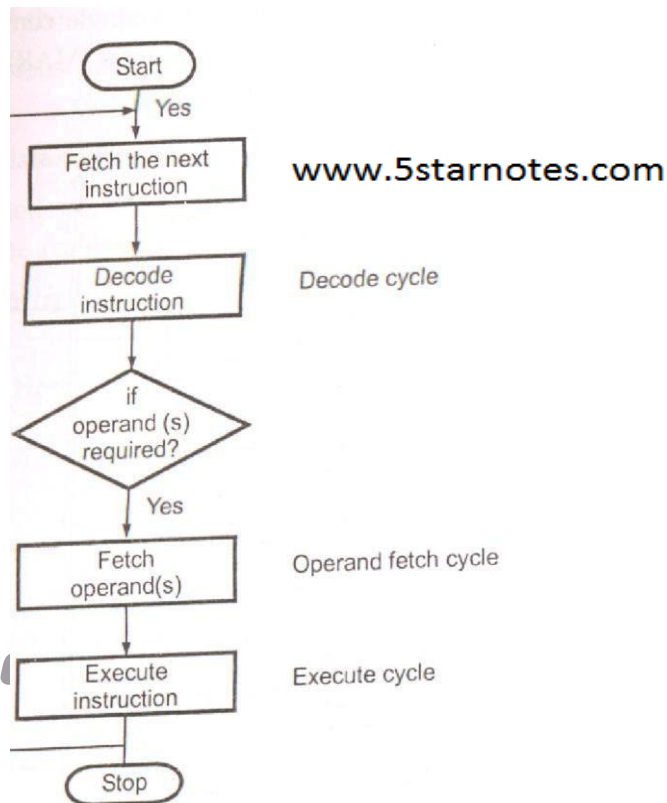
Some fundamental concepts

- The primary function of a processor unit is to execute sequence of instructions stored in a memory, which is external to the processor unit.
- The sequence of operations involved in processing an instruction constitutes an instruction cycle, which can be subdivided into 3 major phases:

1. Fetch cycle
2. Decode cycle
3. Execute cycle

Basic instruction cycle

1.



g. 3.1 Basic instruction cycle

- To perform fetch, decode and execute cycles the processor unit has to perform set of operations called micro-operations.
- Single bus organization of processor unit shows how the building blocks of processor unit are organised and how they are interconnected.

- They can be organised in a variety of ways, in which the arithmetic and logic unit and all processor registers are connected through a single common bus.
- It also shows the external memory bus connected to memory address(MAR) and data register(MDR).

Single Bus Organisation of processor

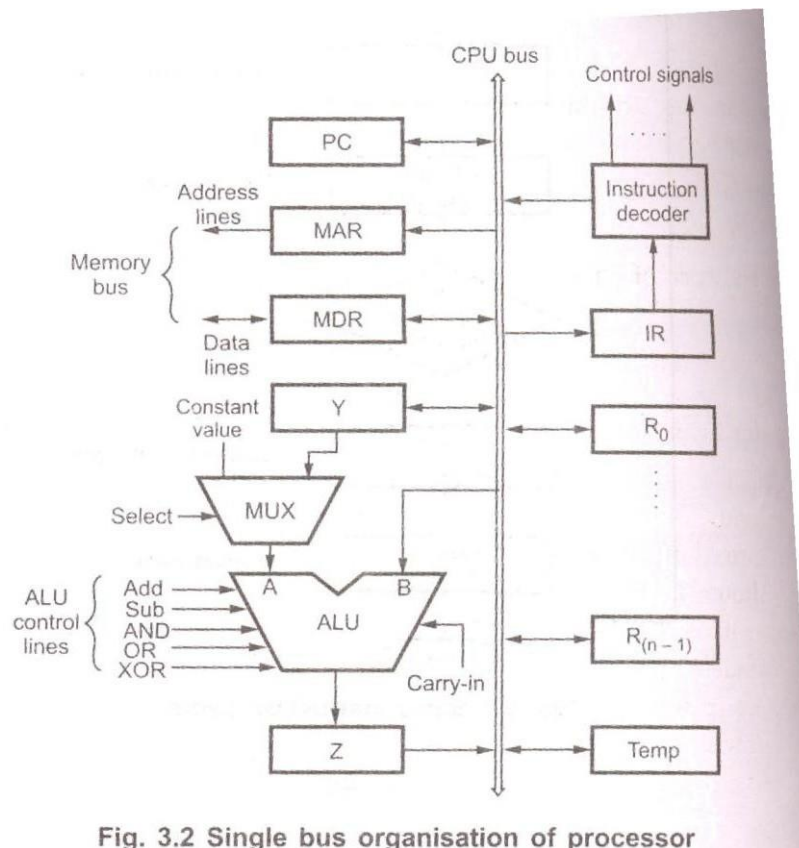


Fig. 3.2 Single bus organisation of processor

- The registers Y,Z and Temp are used only by the processor unit for temporary storage during the execution of some instructions.
- These registers are never used for storing data generated by one instruction for later use by another instruction.
- The programmer cannot access these registers.
- The IR and the instruction decoder are integral parts of the control circuitry in the processing unit.
- All other registers and the ALU are used for storing and manipulating data.
- The data registers, ALU and the interconnecting bus is referred to as data path.
- Register R₀ through R_(n-1) are the processor registers.
- The number and use of these register vary considerably from processor to processor.

- These registers include general purpose registers and special purpose registers such as stack pointer, index registers and pointers.
- These are 2 options provided for A input of the ALU.
- The multiplexer(MUX) is used to select one of the two inputs.
- It selects either output of Y register or a constant number as an A input for the ALU according to the status of the select input.
- It selects output of Y when select input is 1 (select Y) and it selects a constant number when select input is 0(select C) as an input A for the multiplier.
- The constant number is used to increment the contents of program counter.
- For the execution of various instructions processor has to perform one or more of the following basic operations:
 - a) Transfer a word of data from one processor register to the another or to the ALU.
 - b) perform the arithmetic or logic operations on the data from the processor registers and store the result in a processor register.
 - c) Fetch a word of data from specified memory location and load them into a processor register.
 - d) Store a word of data from a processor register into a specified memory location.

1. Register Transfers

Each register has input and output gating and these gates are controlled by corresponding control signals.

Fig: Input and Output Gating for the Registers

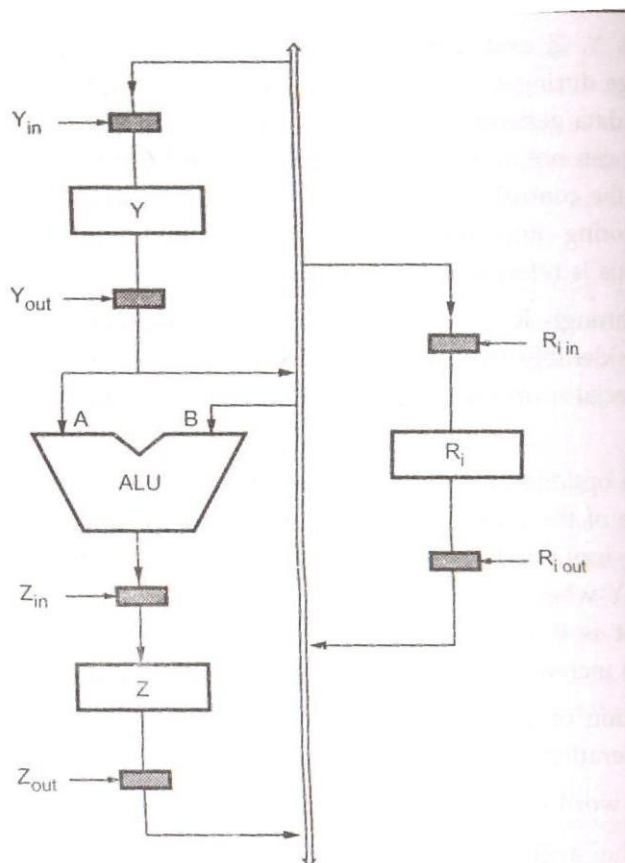


Fig. 3.3 Input and output gating for the register

- The input and output gates are nothing but the electronic switches which can be controlled by the control signals.
- When signal is 1, the switch is ON and when the signal is 0, the switch is OFF.

Implementation of input and output gates of a 4 bit register

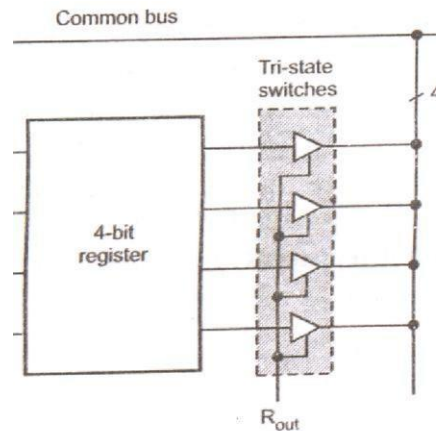


Fig. 3.4

Consider that we have transfer data from register R_1 to R_2

It can be done by,

- Activate the output enable signal of R_1 , $R_1 \text{ out}=1$. It places the contents of R_1 on the common bus.
- Activate the input enable signal of R_2 , $R_2 \text{ in}=1$. It loads data from the common bus into the register R_2 .

One-bit register

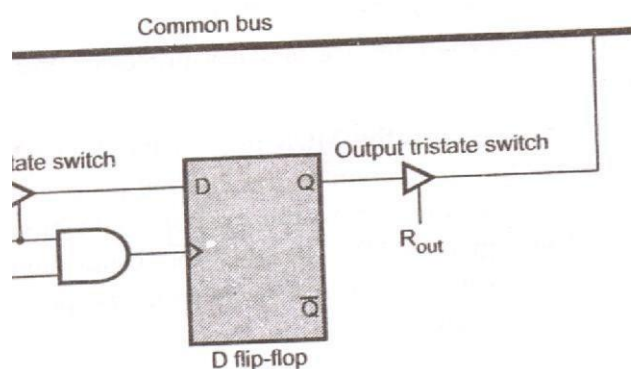


Fig. 3.5 One-bit register

- The edge triggered D flip-flop which stores the one-bit data is connected to the common bus through tri-state switches.
- Input D is connected through input tri-state switch and output Q is connected through output tri-state switch.
- The control signal R_{in} enables the input tri-state switch and the data from common bus is loaded into the D flip-flop in synchronisation with clock input when R_{in} is active.
- It is implemented using AND gate .
- The control signal R_{out} is activated to load data from Q output of the D flip-flop on to the common bus by enabling the output tri-state switch.

3. Performing an arithmetic or logic operation

- ALU performs arithmetic and logic operations.
- It is a combinational circuit that has no internal memory.
- It has 2 inputs A and B and one output.
- It's A input gets the operand from the output of the multiplexer and its B input gets the operand directly from the bus.
- The result produced by the ALU is stored temporarily in register Z.

Let us find the sequence of operations required to subtract the contents of register R_2 from register R_1 and store the result in register R_3 .

This sequence can be followed as:

- a) R_1, Y_{in}
- b) $R_{2out}, \text{Select Y, sub}, Z_{in}$
- c) Z_{out}, R_{3in}

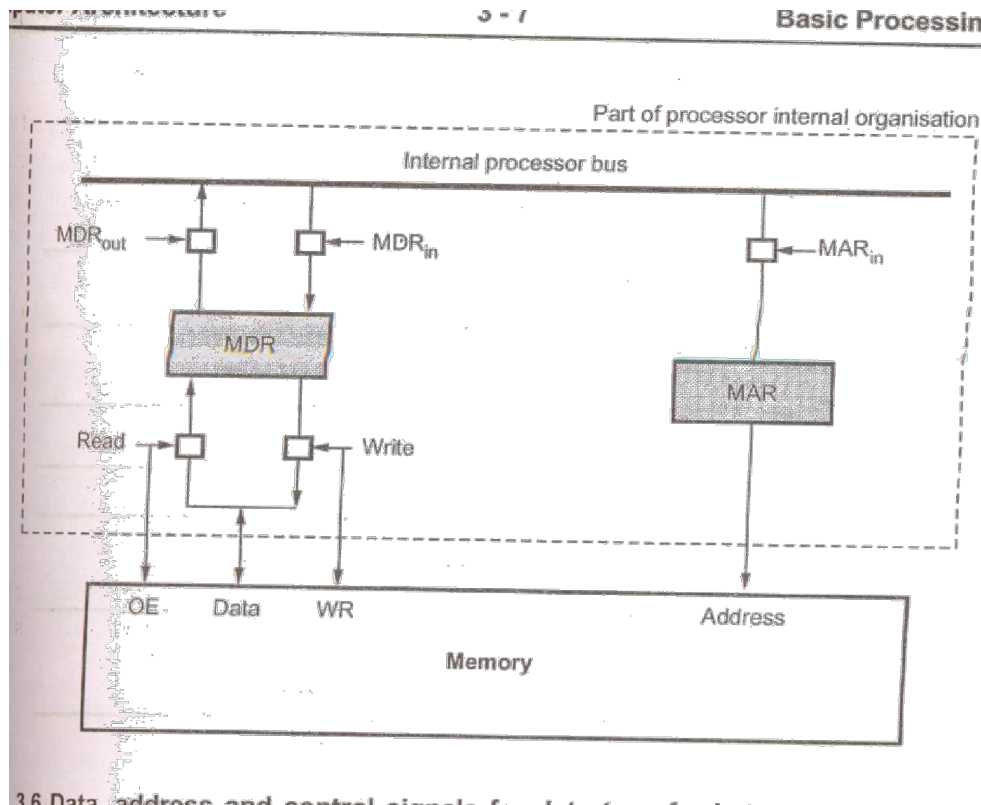
Step 1: contents from register R_1 are loaded into register Y.

Step 2: contents from Y and from register R_2 are applied to the A and B inputs of ALU, subtraction is performed and result is stored in the Z register.

Step 3: The contents of Z register is stored in the R_3 register.

3. Fetching a word from memory

- To fetch a word of data from memory the processor gives the address of the memory location where the data is stored on the address bus and activates the Read operation.
- The processor loads the required address in MAR, whose output is connected to the address lines of the memory bus.
- At the same time processor sends the Read signal of memory control bus to indicate the Read operation.
- When the requested data is received from the memory its stored into the MDR, from where it can be transferred to other processor registers.



4. Storing a word in memory

- To write a word in memory location processor has to load the address of the desired memory location in the MAR, load the data to be written in memory, in MDR and activate write operation.
- Assume that we have to execute instruction Move(R₂), R₁.
- This instruction copies the contents of register R₁ into the memory whose location is specified by the contents of register R₂.

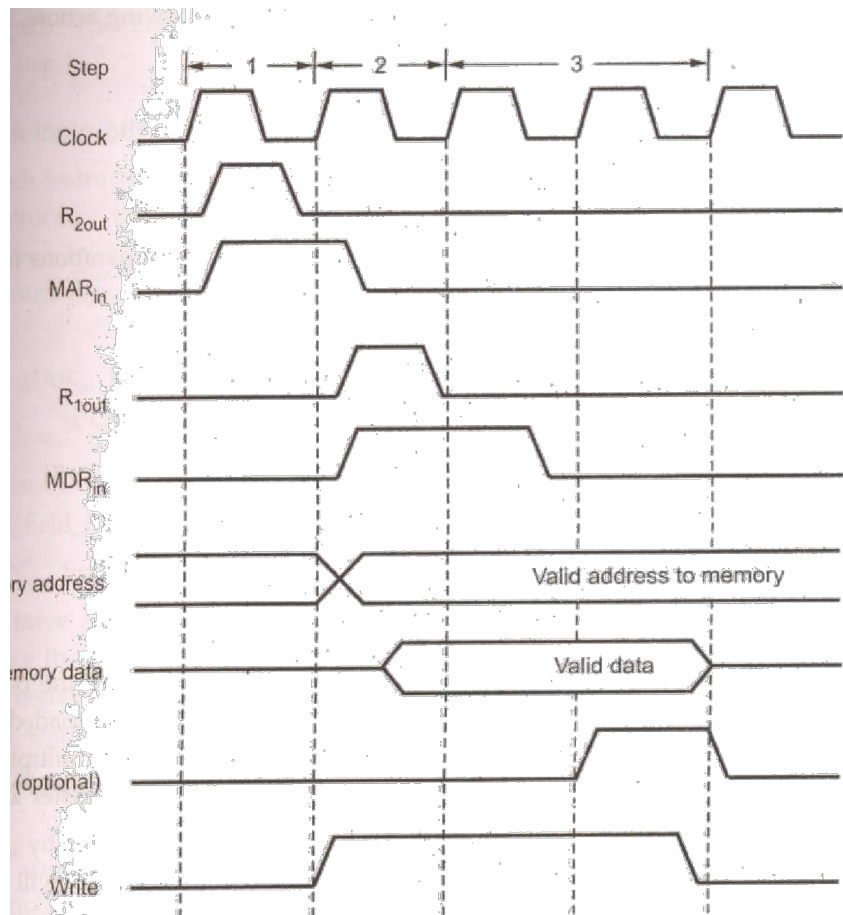
The actions needed to execute this instruction are as follows:

- a) $MAR \leftarrow [R_2]$
- b) $MDR \leftarrow [R_1]$
- c) Activate the control signal to perform the write operation.

The various control signals which are necessary to activate to perform given actions in each step.

- a) R₂_{out}, MAR_{in}
- b) R₁_{out}, MDR_{in}
- c) MAR_{out}, MDR_{out}, Write

Timing diagram for MOVE(R₂), R₁ instruction (Memory write operation)



Timing diagram for `MOVE (R2), R1` instruction (Memory write operation)

- The MDR register has 4 control signals:
 MDR_{inP} & MDR_{outP} control the connection to the internal processor data bus and signals MDR_{inM} & MDR_{outM} control the connection to the memory Data bus.
- MAR register has 2 control signals.
Signal MAR_{in} controls the connection to the internal processor address bus and signal MAR_{out} controls the connection to the memory address bus.
- Control signals read and write from the processor controls the operation Read and Write respectively.

- The address of the memory word to be read word from that location to the register R₃..
- It can be indicated by instruction MOVE R₃,(R₂).

The actions needed to execute this instruction are as follows:

- a) $MAR \leftarrow [R_2]$
- b) Activate the control signal to perform the Read operation
- c) Load MDR from the memory bus
- d) $R_3 \leftarrow MDR$

Various control signals which are necessary to activate to perform given actions in each step:

- a) R_{2out}, MAR_{in}
- b) MAR_{out}, MDR_{inM}, Read
- c) MDR_{outP}, R_{3in}

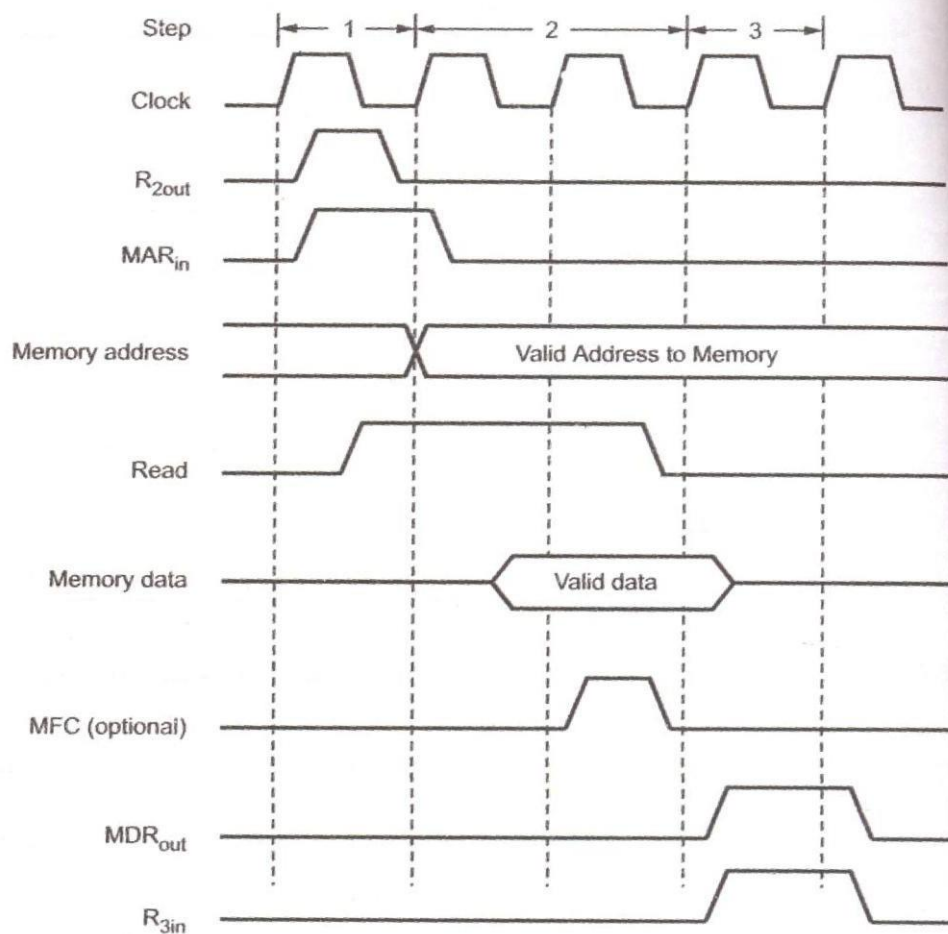


Fig. 3.7 Timing diagram for MOVE R₃, (R₂) instruction (memory read operation)

EXECUTION OF A COMPLETE INSTRUCTION

Let us find the complete control sequence for execution of the instruction Add $R_1, (R_2)$ for the single bus processor.

- This instruction adds the contents of register R_1 and the contents of memory location specified by register R_2 and stores results in the register R_1 .
- To execute bus instruction it is necessary to perform following actions:
 1. Fetch the instruction
 2. Fetch the operand from memory location pointed by R_2 .
 3. Perform the addition
 4. Store the results in R_1 .

The sequence of control steps required to perform these operations for the single bus architecture are as follows;

1. $PC_{out}, MAR_{in}, Y_{in}, \text{select C, Add, } Z_{in}$
2. $Z_{out}, PC_{in}, MAR_{out}, MAR_{inM}, \text{Read}$
3. $MDR_{out} P, MAR_{in}$
4. R_{2out}, MAR_{in}
5. $R_{2out}, Y_{in}, MAR_{out}, MAR_{inM}, \text{Read}$
6. $MDR_{out} P, \text{select Y, Add, } Z_{in}$
7. Z_{out}, R_{1in}

(i) Step1, the instruction fetch operation is initiated by loading the controls of the PC into the MAR.

- PC contents are also loaded into register Y and added constant number by activating select C input of multiplexer and add input of the ALU.
- By activating Z_{in} signal result is stored in the register Z

(iii) Step2, the contents of register Z are transferred to pc register by activating Z_{out} and pc_{in} signal.

- This completes the PC increment operation and PC will now point to next instruction,
- In the same step (step2), MAR_{out} , MDR_{inM} and Read signals are activated.
- Due to MAR_{out} signal, memory gets the address and after receiving read signal and activation of MDR in M Signal, it loads the contents of specified location into MDR register.

(iv) Step 3 contents of MDR register are transferred to the instruction register(IR) of the processor.

- The step 1 through 3 constitute the instruction fetch phase.
- At the beginning of step 4, the instruction decoder interprets the contents of the IR.
- This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase.

(v) Step 4, the contents of register R_2 are transferred to register MAR by activating R_{2out} and MAR_{in} signals.

(v) Step 5, the contents of register R₁ are transferred to register Y by activating R_{1out} and Y_{in} signals. In the same step, MAR_{out}, MDR_{inM} and Read signals are activated.

- Due to MAR_{out} signal, memory gets the address and after receiving read signal and activation of MDR_{inM} signal it loads the contents of specified location into MDR register.

(vi) Step 6 MDR_{outP}, select Y, Add and Z_{in} signals are activated to perform addition of contents of register Y and the contents of MDR. The result is stored in the register Z.

(vii) Step 7, the contents of register Z are transferred to register R₁ by activating Z_{out} and R_{1in} signals.

Branch Instruction

The branch instruction loads the branch target address in PC so that PC will fetch the next instruction from the branch target address.

The branch target address is usually obtained by adding the offset in the contents of PC. The offset is specified within the instruction.

The control sequence for unconditional branch instruction is as follows:

1. PC_{out}, MAR_{in}, Y_{in}, SelectC, Add, Z_{in}
2. Z_{out}, PC_{in}, MAR_{out}, MDR_{inM}, Read
3. MDR_{outP}, IR_{in}
4. PC_{out}, Y_{in}
5. Offset_field_Of_IR_{out}, SelectY, Add, Z_{in}
6. Z_{out}, PC_{in}

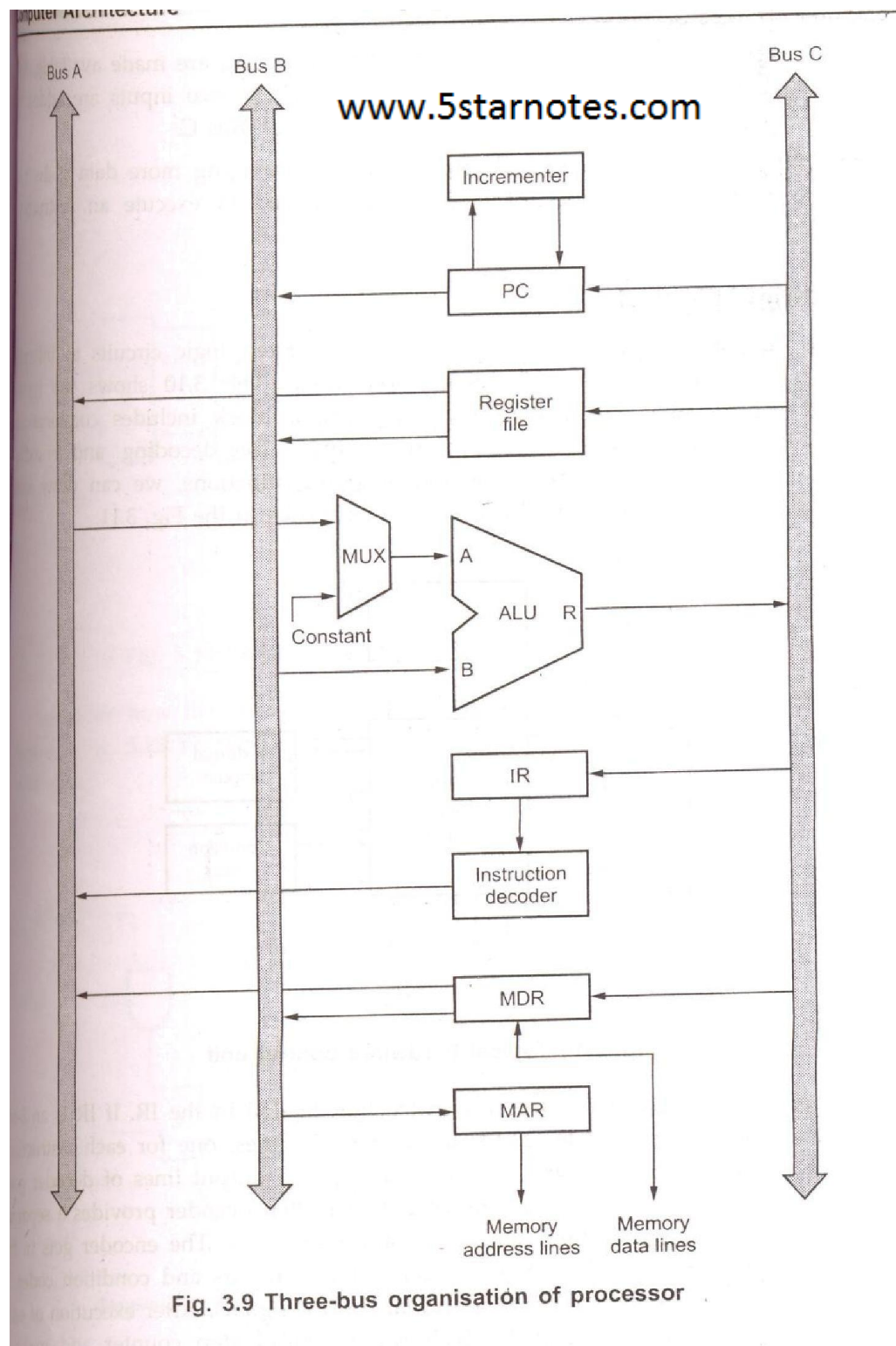
First 3 steps are same as in the previous example.

Step 4: The contents of PC are transferred to register Y by activating PC_{out} and Y_{in} signals.

Step 5: The contents of PC and the offset field of IR register are added and result is saved in register Z by activating corresponding signals.

Step 6: The contents of register Z are transferred to PC by activating Z_{out} and PC_{in} signals.

Multiple Bus Organisation:



- Single bus only one data word can be transferred over the bus in a clock cycle.
- This increases the steps required to complete the execution of the instruction.
- To reduce the number of steps needed to execute instructions, most commercial process provide multiple internal paths that enable several transfer to take place in parallel.
- 3 buses are used to connect registers and the ALU of the processor.
- All general purpose registers are shown by a single block called register file.
- There are 3 ports, one input port and two output ports.
- So it is possible to access data of three register in one clock cycle, the value can be loaded in one register from bus C and data from two register can be accessed to bus A and bus B.
- Buses A and B are used to transfer the source operands to the A and B inputs of the ALU.
- After performing arithmetic or logic operation result is transferred to the destination operand over bus C.
- To increment the contents of PC after execution of each instruction to fetch the next instruction, separate unit is provided. This unit is known as incrementer.
- Incrementer increments the contents of PC accordingly to the length of the instruction so that it can point to next instruction in the sequence.
- The incrementer eliminates the need of multiplexer connected at the A input of ALU.
- Let us consider the execution of instruction, Add,R₁,R₂,R₃.
- This instruction adds the contents of registers R₂ and the contents of register R₃ and stores the result in R₁.
- With 3 bus organization control steps required for execution of instruction Add R₁,R₂,R₃ are as follows:
 1. PC_{out}, MAR_{in}
 2. MAR_{out}, MDR_{in}M, Read
 3. MDR_{out}P,IR_{in}
 4. R₂_{out},R₃_{out},Add,R₁_{in}

Step 1: The contents of PC are transferred to MAR through bus B.

Step 2: The instruction code from the addressed memory location is read into MDR.

Step 3: The instruction code is transferred from MDR to IR register. At the beginning of step 4, the instruction decoder interprets the contents of the IR.

This enables the control circuitry to activate the control signals for step 4, which constitute the execution phase.

Step 4: two operands from register R₂ and register R₃ are made available at A and B inputs of ALU through bus A and bus B.

These two inputs are added by activation of Add signal and result is stored in R₁ through bus C.

Hardwired Control

The control units use fixed logic circuits to interpret instructions and generate control signals from them.

The fixed logic circuit block includes combinational circuit that generates the required control outputs for decoding and encoding functions.

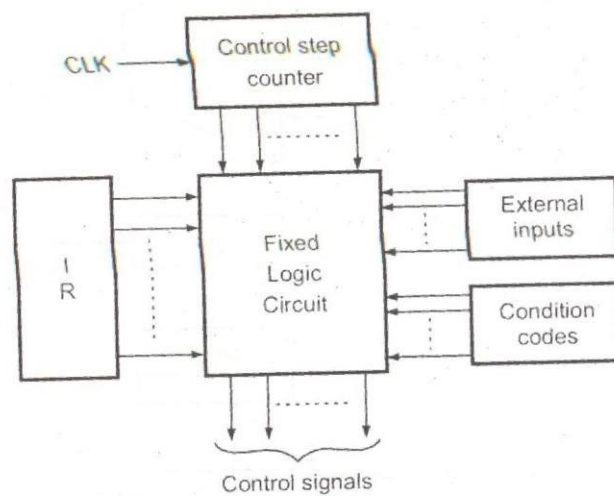


Fig. 3.10 Typical hardwired control unit

Decoder decodes the instruction loaded in the IF

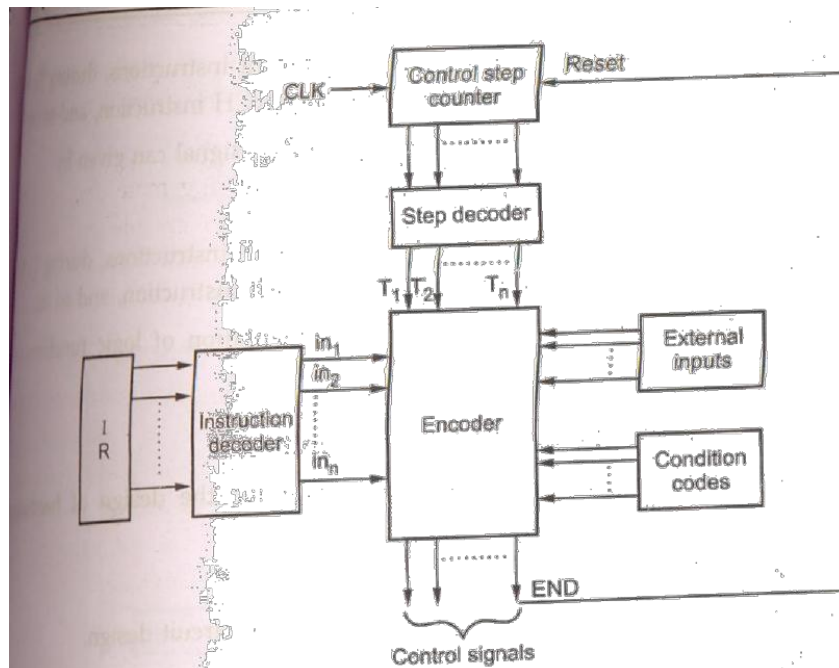


Fig. 3.11 Detail block diagram for hardwired control unit

Let us see how the encoder generates signal for single bus processor organisation shown in Fig. 3.12 Y_{in} . The encoder circuit implements the following logic function to generate Y_{in} .

$$Y_{in} = T_1 + T_5 \cdot \text{ADD} + T_4 \cdot \text{BRANCH} + \dots$$

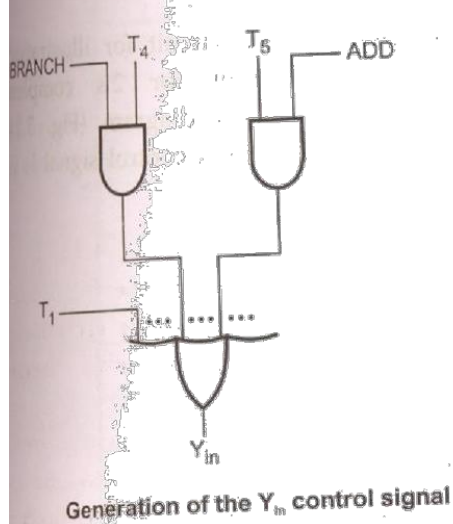


Fig. 3.12

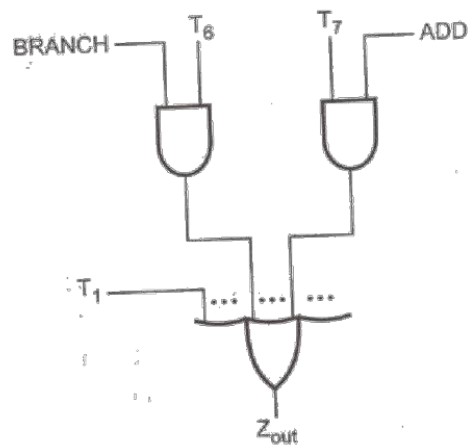


Fig. 3.13

Instruction decoder

It decodes the instruction loaded in the IR.

If IR is an 8 bit register then instruction decoder generates 2^8 (256 lines); one for each instruction.

According to code in the IR, only one line amongst all output lines of decoder goes high (set to 1 and all other lines are set to 0).

Step decoder

It provides a separate signal line for each step, or time slot, in a control sequence.

Encoder

It gets in the input from instruction decoder, step decoder, external inputs and condition codes.

It uses all these inputs to generate the individual control signals.

After execution of each instruction end signal is generated this resets control step counter and make it ready for generation of control step for next instruction.

The encoder circuit implements the following logic function to generate Y_{in}

$$Y_{in} = T_1 + T_5 \cdot \text{ADD} + T_4 \cdot \text{BRANCH} + \dots$$

The Y_{in} signal is asserted during time interval T_1 for all instructions, during T_5 for an ADD instruction, during T_4 for an unconditional branch instruction, and so on.

As another example, the logic function to generate Z_{out} signal can given by

$$Z_{out} = T_2 + T_7 \cdot \text{ADD} + T_6 \cdot \text{BRANCH} + \dots$$

The Z_{out} signal is asserted during time interval T_2 of all instructions, during T_7 for an ADD instruction, during T_6 for an unconditional branch instruction, and so on.

A Complete processor

It consists of

- Instruction unit
- Integer unit
- Floating-point unit
- Instruction cache
- Data cache
- Bus interface unit
- Main memory module
- Input/Output module.

Instruction unit- It fetches instructions from an instruction cache or from the main memory when the desired instructions are not available in the cache.

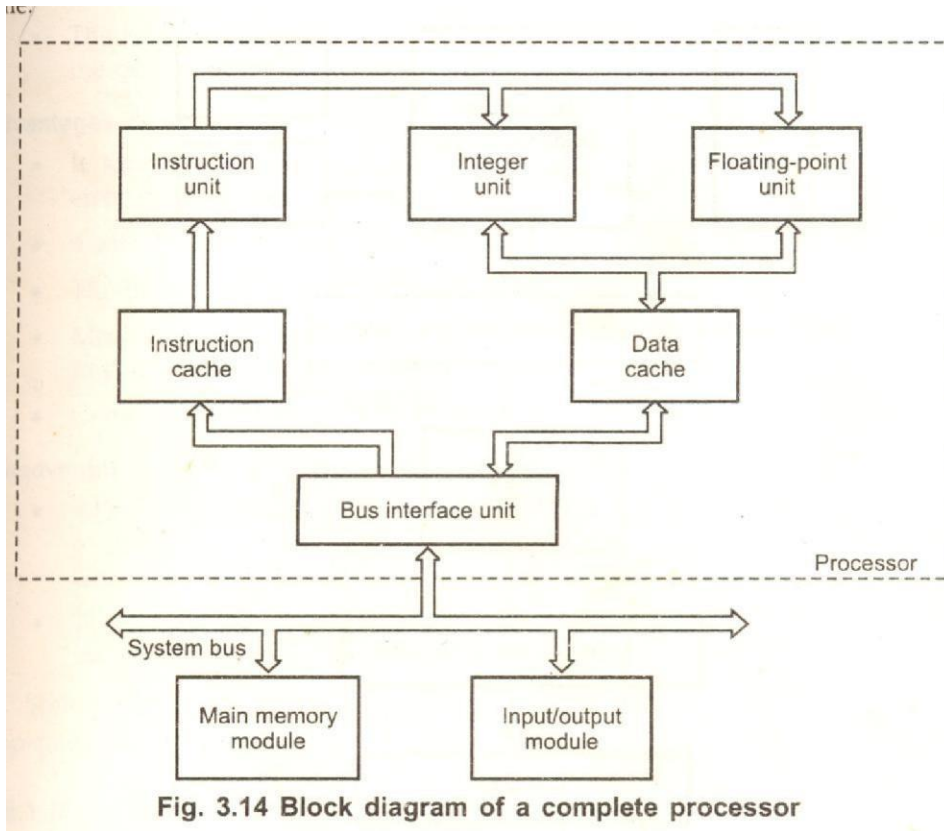
Integer unit – To process integer data

Floating unit – To process floating –point data

Data cache – The integer and floating unit gets data from data cache

The 80486 processor has 8-kbytes single cache for both instruction and data whereas the Pentium processor has two separate 8 kbytes caches for instruction and data.

The processor provides bus interface unit to control the interface of processor to system bus, main memory module and input/output module.



Microprogrammed Control

Every instruction in a processor is implemented by a sequence of one or more sets of concurrent micro operations.

Each micro operation is associated with a specific set of control lines which, when activated, causes that micro operation to take place.

Since the number of instructions and control lines is often in the hundreds, the complexity of hardwired control unit is very high.

Thus, it is costly and difficult to design. The hardwired control unit is relatively inflexible because it is difficult to change the design, if one wishes to correct design error or modify the instruction set.

Microprogramming is a method of control unit design in which the control signal memory CM.

The control signals to be activated at any time are specified by a microinstruction, which is fetched from CM.

A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program.

The micro programs for all instructions are stored in the control memory.

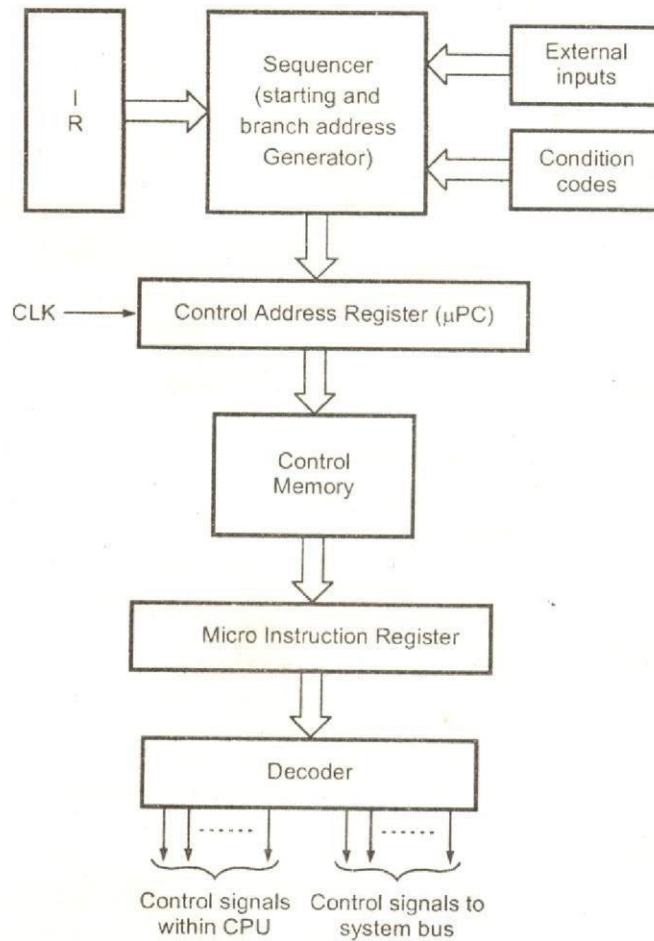


Fig. 3.15 Microprogrammed control unit

The address where these microinstructions are stored in CM is generated by microprogram sequencer/microprogram controller.

The microprogram sequencer generates the address for microinstruction according to the instruction stored in the IR.

The microprogrammed control unit,

- control memory
- control address register
- micro instruction register
- microprogram sequencer

The components of control unit work together as follows:

- The control address register holds the address of the next microinstruction to be read.
- When address is available in control address register, the sequencer issues READ command to the control memory.
- After issue of READ command, the word from the addressed location is read into the microinstruction register.
- Now the content of the micro instruction register generates control signals and next address information for the sequencer.
- The sequencer loads a new address into the control address register based on the next address information.

Advantages of Microprogrammed control

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

Microinstruction

A simple way to structure microinstructions is to assign one bit position to each control signal required in the CPU.

Grouping of control signals

Grouping technique is used to reduce the number of bits in the microinstruction.

Gating signals: IN and OUT signals

Control signals: Read, Write, clear A, Set carry in, continue operation, end, etc. ALU signals: Add, Sub, etc;

There are 46 signals and hence each microinstruction will have 46 bits.

It is not at all necessary to use all 46 bits for every microinstruction because by using grouping of control signals we minimize number of bits for microinstruction.

Way to reduce number of bits microinstruction:

- Most signals are not needed simultaneously.
- Many signals are mutually exclusive
e.g. only one function of ALU can be activated at a time.
- A source for data transfers must be unique which means that it should not be possible to get the contents of two different registers on to the bus at the same time.
- Read and Write signals to the memory cannot be activated simultaneously.

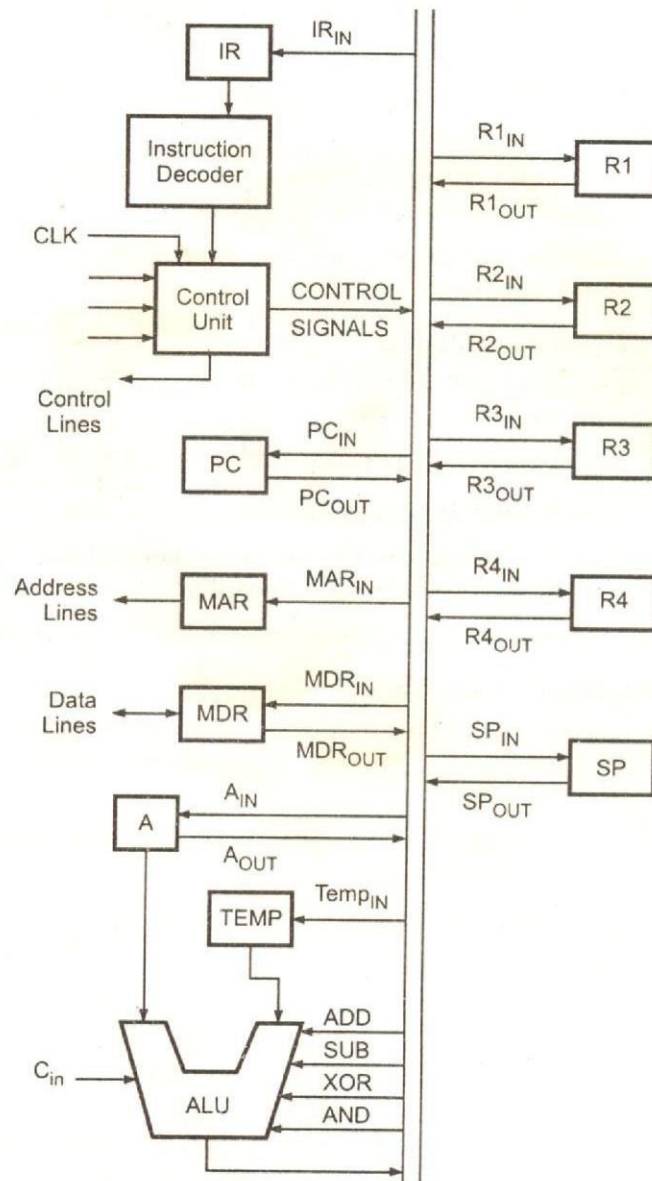


Fig. 3.16 Single bus CPU structure with control signals

46 control signals can be grouped in 7 different groups.

G ₁ (4 Bits) : IN grouping		G ₂ (4 Bits) : OUT grouping	
0000	No Transfer	0000	No Transfer
0001	IR _{IN}	0001	PC _{OUT}
0010	PC _{IN}	0010	MDR _{OUT}
0011	MDR _{IN}	0011	R _{1OUT}
0100	MAR _{IN}	0100	R _{2OUT}
0101	A _{IN}	0101	R _{3OUT}
0110	Temp _{IN}	0110	R _{4OUT}
0111	R _{1IN}	0111	SP _{OUT}
1000	R _{2IN}		
1001	R _{3IN}		
1010	R _{4IN}		
1011	SP _{IN}		

G ₃ (4 bits) : ALU Functions		G ₄ (2 Bits) : RD/WR Control Signals	
0000	ADD	00	No Action
0001	SUB	01	Read
...	16 ALU Functions	10	Write
...			
1111	XOR		

G ₅ (1 Bit) : A Register	G ₆ (1 Bit) : Carry
0 - No action	0 - Carry in to ALU = 0
1 - Clear A	1 - Carry in to ALU = 1
G ₇ (1 bit) :	G ₈ (1 bit) : Operation
0 : No action	0 : Continue Operation
1 : WMFC	1 : End

The total number of grouping bits are 17. Therefore, we minimized 46 bits microinstruction to 17 bit microinstruction.

Techniques of grouping of control signals

The grouping of control signal can be done either by using technique called vertical organisation or by using technique called vertical organisation or by using technique called horizontal organisation.

Vertical organisation

Highly encoded scheme that can be compact codes to specify only a small number of control functions in each microinstruction are referred to as a vertical organisation.

Horizontal organisation

The minimally encoded scheme, in which resources can be controlled with a single instruction is called a horizontal organisation.

Comparison between horizontal and vertical organisation

S.No	Horizontal	Vertical
1	Long formats	Short formats
2	Ability to express a high degree of parallelism	Limited ability to express parallel microoperations
3	Little encoding of the control information	Considerable encoding of the control information
4	Useful when higher operating speed is desired	Slower operating speeds

Advantages of vertical and horizontal organization

1. Vertical approach is the significant factor, it is used to reduce the requirement for the parallel hardware required to handle the execution of microinstructions.
2. Less bits are required in the microinstruction.
3. The horizontal organisation approach is suitable when operating speed of computer is a critical factor and where the machine structure allows parallel usage of a number of resources.

Disadvantages

Vertical approach results in slower operations speed.

Microprogram sequencing

The task of microprogram sequencing is done by microprogram sequencer.

2 important factors must be considered while designing the microprogram sequencer:

- a) The size of the microinstruction
- b) The address generation time.

The size of the microinstruction should be minimum so that the size of control memory required to store microinstructions is also less.

This reduces the cost of control memory.

With less address generation time, microinstruction can be executed in less time resulting better throughput.

During execution of a microprogram the address of the next microinstruction to be executed has 3 sources:

- i. Determined by instruction register
- ii. Next sequential address
- iii. Branch

- Microinstructions can be shared using microinstruction branching.

Consider instruction ADD src, Rdst.

- The instruction adds the source operand to the contents of register Rdst and places the sum in Rdst, the destination register.

Let us assume that the source operand can be specified in the following addressing modes:

- Indexed
- Autoincrement
- Autodecrement
- Register indirect
- Register direct

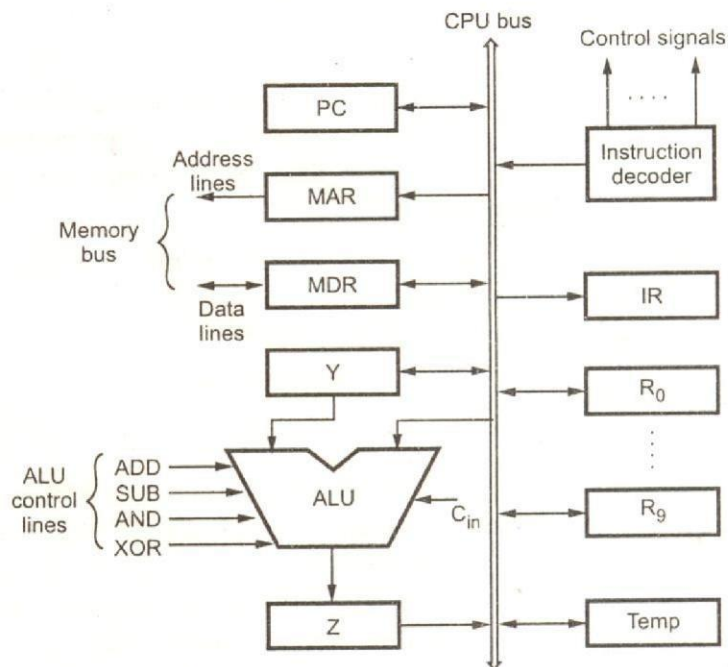


Fig. 3.17 CPU structure

Each box in the flowchart corresponds to a microinstruction that controls the transfers and operations indicated within the box.

The microinstruction is located at the address indicated by the number above the upper right-hand corner of the box.

During the execution of the microinstruction, the branching takes place at point A. The branching address is determined by the addressing mode used in the instruction.

Techniques for modification or generation of branch addresses

i. Bit-ORing

The branch address is determined by ORing particular bit or bits with the current address of microinstruction.

Eg: If the current address is 170 and branch address is 172 then the branch address can be generated by ORing 02(bit 1), with the current address.

i. Using condition variables

It is used to modify the contents CM address register directly, thus eliminating whole or in part the need for branch addresses in $\mu\mu\mu\mu\mu$ microinstructions.

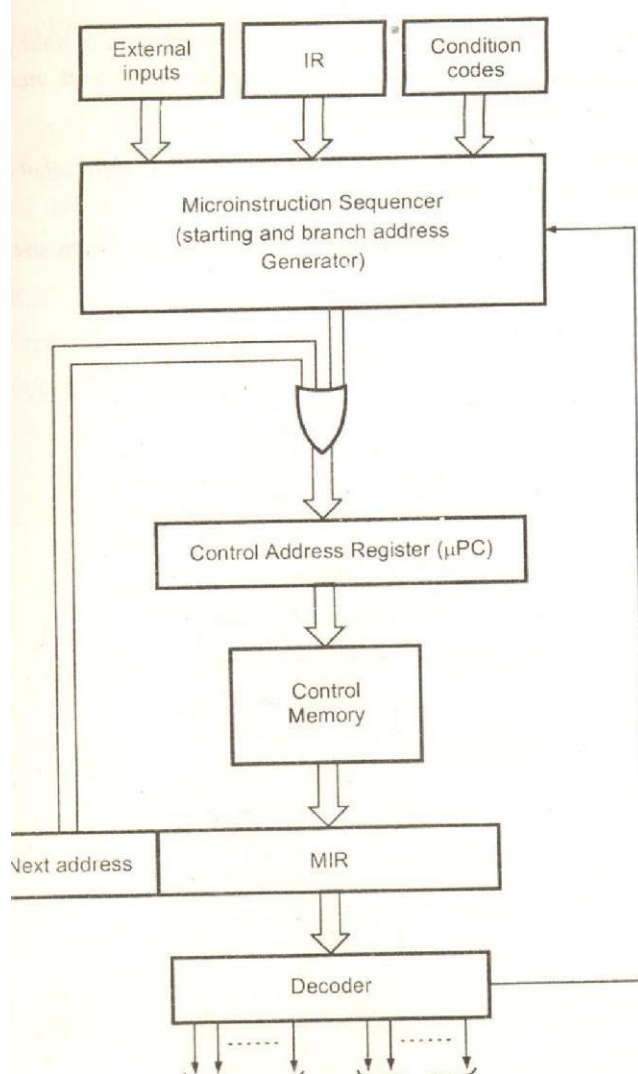
Eg: Let the condition variable CY indicate occurrence of CY = 1, and no carry when CY = 0.

Suppose that we want to execute a SKIP_ON_CARRY microinstruction.

It can be done by logically connecting CY to the count enable input of μpc at an appropriate point in the microinstruction cycle. It allows the overflow condition increment μpc an extra time, thus performing the desired skip operation.

iii. Wide-Branch Addressing

Generating branch addresses becomes more difficult as the number of branches increases. In such situations programmable logic array can be used to generate the required branch addresses. The simple and inexpensive way of generating branch addresses is known as wide-branch addressing. The opcode of a machine instruction is translated into the starting address of the corresponding micro-routine. This is achieved by connecting the opcode bits of the instruction register as inputs to the PLA, which acts as a decoder. The output of the PLA is the address of the desired micro-routine.



Comparison between Hardwired and Microprogrammed Control

Attribute	Hardwired Control	Microprogrammed Control
Speed	Fast	Slow
Control functions	Implemented in hardware	Implemented in software
Flexibility	Not flexible to accommodate new system specifications or new instructions	More flexible, to accommodate new system specification or new instructions redesign is required
Ability to handle large/complex instruction sets	Difficult	Easier
Ability to support operating systems and diagnostic features	Very difficult	Easy
Design process	Complicated	Orderly and systematic
Applications	Mostly RISC microprocessors	Mainframes, some microprocessors
Instructionset size	Usually under 100 instructions	Usually over 100 instructions
ROM size	-	2K to 10K by 20-400 bit microinstructions
Chip area efficiency	Uses least area	Uses more area