3 ALU, Data-path & Control Unit

Introduction:

- A computer is used for the execution of the program, which consists of a set of instructions stored in the memory.
- The actual work of executing the instructions of the program is done by the processor(CPU).
- To process an instruction, CPU uses three internal components named as:
  1) Registers
  2) ALU (arithmetic and logic unit)
  3) Control unit (CU)

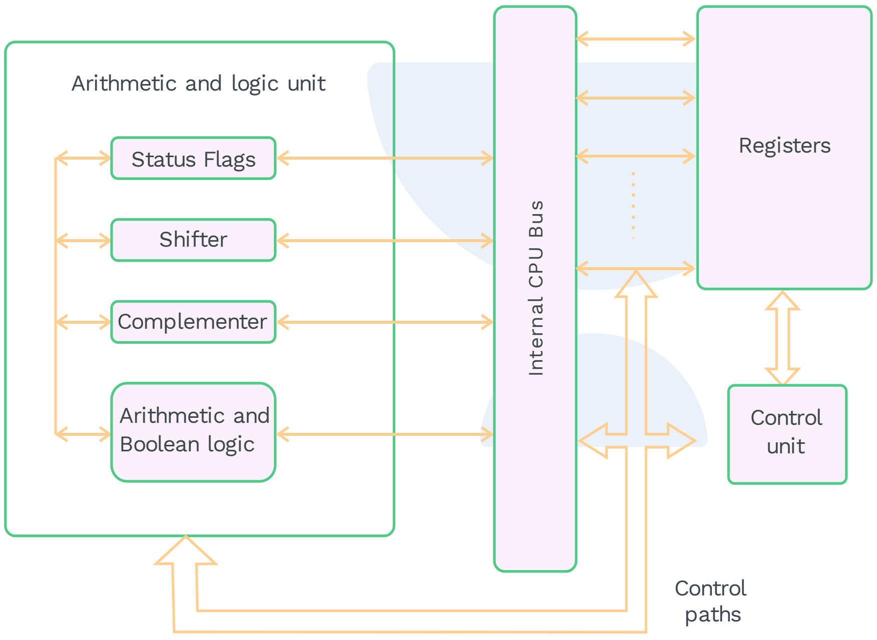


Fig. 3.1 Internal Structure of CPU

1) Register:

- A register is a storage component present inside the CPU which is used to accept, store and transfer the data and instructions used by the CPU.
- Registers are very fast computer memory units because they are made up of a group of flip flops and gates.
- Flip Flops are used to store the binary information, and gates control the transfer of information in and out of registers.

2) ALU (arithmetic and logic unit):

- The arithmetic and logical operations on data items inside the computer are performed by ALU.

- All the other components like control unit, registers, memory, and I/O bring the data for processing into the ALU and then take the result back out.
- Data to be processed is forwarded to registers and ALU, and the result of ALU operation is stored in specified registers.
- Based on the computation result, the ALU can also set flags.
  For example, if the result of an ALU operation which is to be stored in a register exceeds the length of that register, then overflow is set to 1.
- The control unit sends the signals that control the operation of the ALU and also controls the movement of the data into and out of the ALU.

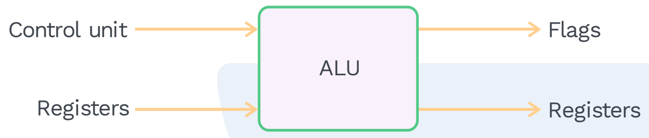


Fig. 3.2 ALU Inputs and Outputs

3) Control unit (CU):

- Control unit is that component of the processor that causes things to happen, i.e. it directs the operation of the processor.
- Control unit converts the information that is being received into the control signals and then transfers the control signals to the central processor.
- This processor then tells the hardware what operations are to be performed based on the control signals.
- Control unit consists of inputs such as instruction registers, flags and control signals from external sources (e.g. interrupt signals).

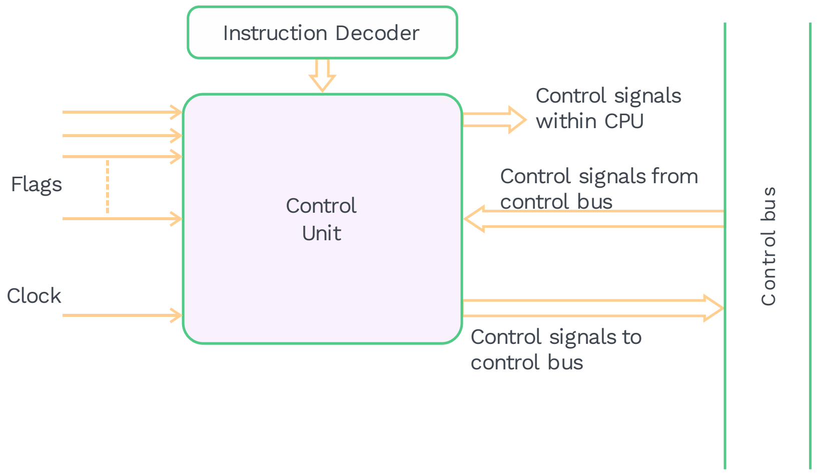

Instruction Decoder
Control signals within CPU
Control signals from control bus
Flags
Control Unit
Clock
Control signals to control bus
Control bus

Fig. 3.3 Block Diagram of Control Unit

Control unit functions:

- In order to exchange data between memory and I/O modules, control signals are being issued by the control unit external to the processor.
- Control signals internal to the processor are being issued by the control unit for ALU to perform the specified operation, move the data between the registers and to regulate other internal operations.
- Control unit handles tasks such as fetching, decoding, executing and storage of results of the micro-operations.
- It regulates that the data inside the processor will be executed in what sequence and will be stored in which register, i.e. it directs the data flow sequence inside the processor as well as outside the processor.
- The two basic tasks performed by the control unit are:
  1) **Sequencing:** The control unit makes the processor move through a series of micro-operations in the proper sequence based on which the program is being executed.
  2) **Execution:** The control unit is also responsible for each micro-operation to be performed.

Micro-operations:

- Execution of a program comprises sequential execution of instructions.
- Execution of each instruction is done during an instruction cycle which is made up of shorter sub cycles (e.g. fetch cycle, execute cycle, interrupt cycle).
- Execution of each sub cycle consists of one or more shorter operations, called micro-operations.
- Micro-operations are the functional or atomic operations of a processor that finishes its execution in 1 clock cycle.

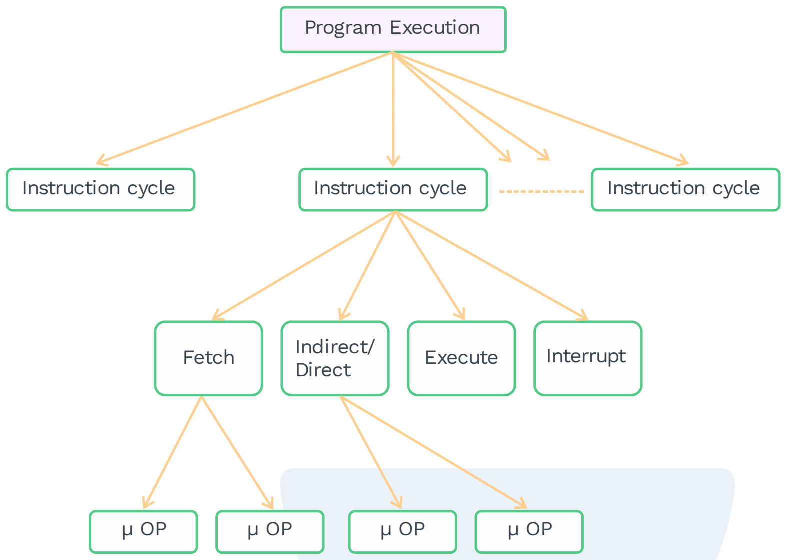


Fig. 3.4 Constituent Elements of Execution of Program

Note:

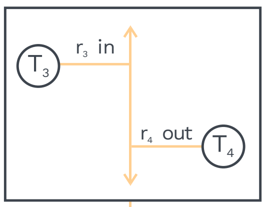
Register to Register transfer operation is a μ-operation from the user's point of view.

Example: Following is the micro-operation to move the content from one register to another which will be completed in one clock cycle.

Mov r_3, r_4

Register transfer language : $r_3 \leftarrow r_4$

H/W Design :



μ-operation list : $\underbrace{T_1 : r_{4out}, r_{3in}}_{\text{Control signals}}$

Micro-instruction:

Note:

Micro-instruction designed with one micro-operation or more than 1 μ-operation, will always takes 1 cycle to complete.

Definitions

Micro-instruction is a collection of micro-operations.

Micro-program:

- Sequence of μ-instructions which are used to execute a task in the hardware is known as μ-program.

Example: In order to execute an instruction, the instruction must be brought to the processor from the memory. To complete this task following micro-program is used:

t_1: MAR ← (PC)

t_2: MBR ← Memory

PC ← (PC) + step size

t_3: IR ← (MBR)

The above micro-program consists of four micro-instructions, which takes three clock cycles.

The fetch cycle:

- Every instruction is placed in the main memory.
- In order to execute an instruction, it must be brought into the CPU from the main memory.
- Thus, the fetch cycle appears at the starting of every instruction cycle and causes an instruction to be fetched from memory.
- Following are the sequence of events for the fetch cycle:

1) Program Counter(PC) consist of the address of the next instruction which is to be executed, and this address is moved to the memory address register (MAR).

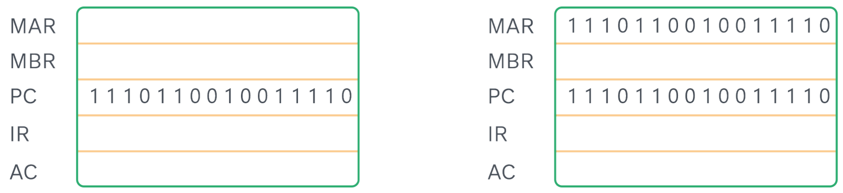


Fig. 3.5 Orientation of PC and MAR

2) Next step is to bring the instruction from the memory.

The MAR contains the instruction address, which is placed on the address bus and control unit sends the READ command on the control bus, and the fetched instruction appears on the data bus, which is then copied to memory buffer register (MBR)

MAR	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0
MBR	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0
PC	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1
IR	
AC	

Fig. 3.6 Orientation of MAR and MBR

Since read from memory and incrementing PC to the instruction size does not affect each other, so in this step, PC (program counter) can also be incremented.

3) Now, the data is placed from the MBR to IR.

MAR	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0
MBR	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0
PC	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1
IR	1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0
AC	

Fig. 3.7 Orientation of MBR and IR

Thus, the fetch cycle has four micro operations, which can be completed in three clock cycles where each micro operation involves movement of data in and out of registers.

Hardware design:

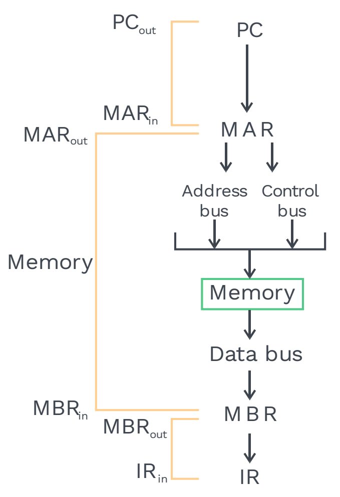


Fig. 3.8 Hardware Design of IF Cycle

IF μ-program:

t_1: MAR ← (PC)

t_2: MBR ← Memory

PC ← (PC) + I

t_3: IR ← (MBR)

Where I is the instruction length and t_1, t_2, and t_3 are the time units.

First time unit (t_1):	MAR_{in}, PC_{out}
Second time unit (t_2):	MAR_{out}, Memory, MBR_{in}
	PC_{out}, Increment, PC_{in}
Third time unit (t_3):	MBR_{out}, IR_{in}

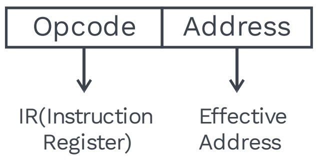
The operand fetch cycle:

- After the instruction is fetched, the source operands must also be fetched.
- .If the instruction specifies a direct/indirect address, then a direct/indirect cycle must precede accordingly before the execute cycle.

a) Direct addressing mode:

Let us assume a one-address instruction format with direct addressing mode.

Instruction:



- During the instruction fetch, the instruction is placed in the IR(Instruction Register).
- The address field of the instruction is moved to the memory address register (MAR).
- Then, this address is used to fetch the operand from the memory that is stored in the corresponding address.
- The operand which is fetched from memory is kept on the data bus and is then placed in the memory buffer register (MBR).
- The memory buffer register (MBR) places the operand in the accumulator register for further processing.

Hardware design:

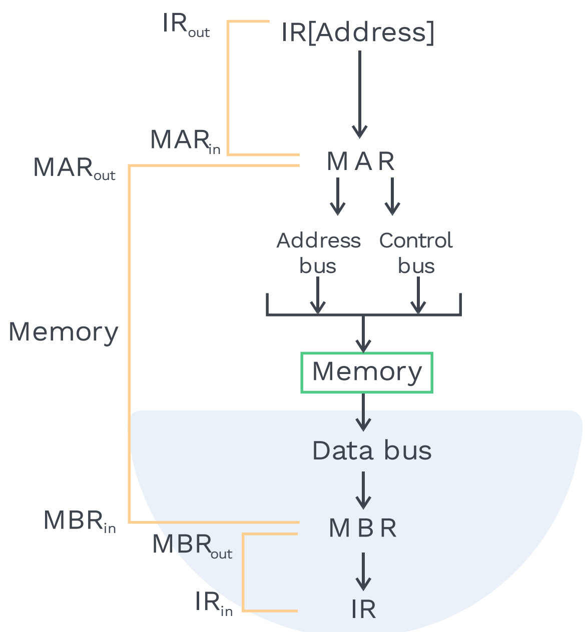


Fig. 3.9 Hardware Design of Direct Operand Fetch Cycle

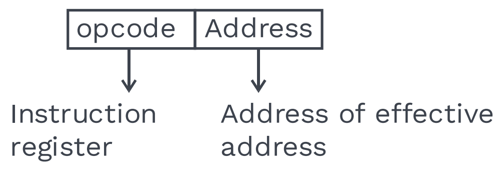
Micro-program of direct operand fetch cycle:

t_1: MAR $\leftarrow$ IR[Address]

t_2: MBR $\leftarrow$ M[MAR]

t_3: AC $\leftarrow$ MBR

b) Indirect addressing mode:
An example of a one-address instruction format that uses indirect addressing mode is considered for the detailed analysis below.
Instruction:



- The address field of the instruction placed in the instruction register (IR) is transferred to the memory address register (MAR).
- The operand's effective address is achieved by accessing the memory address mentioned in the address field of the given instruction format.
- The effective address(EA) which is being fetched is placed in the MBR and the address field of the instruction register is replaced by

the effective address(EA), so now, instead of an indirect address, the address field contains a direct address.

- Then again, the address field of IR is transferred to MAR, and this address is used to fetch the operand from the corresponding memory location.
- The fetched operand appears on the data bus and is then placed in MBR.
- Then, from the memory buffer register (MBR), the operand is placed in accumulator register(AC) for further processing / execution.

Hardware design:

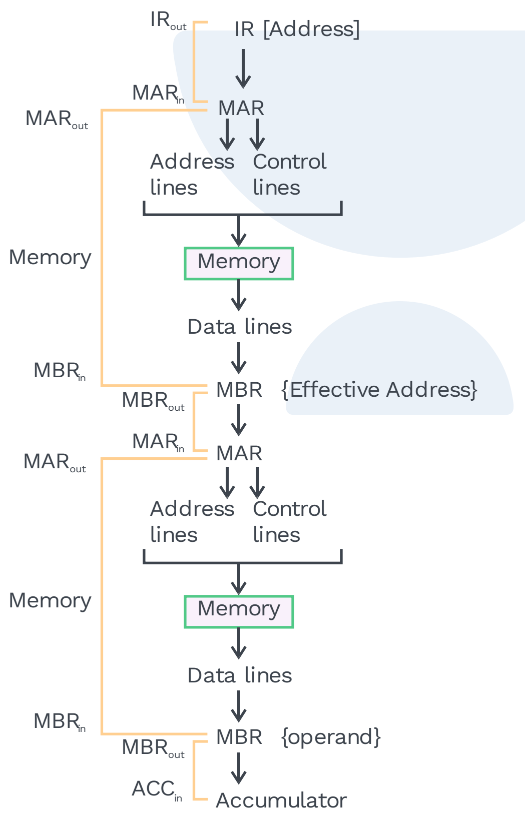


Fig. 3.10 Hardware Design of Indirect Operand Fetch Cycle

Microprogram of indirect operand fetch cycle:

t_1: MAR ← (IR (address))

t_2: MBR ← Memory

t_3: MAR ← MBR

t_4: MBR ← Memory

t_5: Accumulator ← MBR

The execute cycle:

- The micro-operations in the execution cycle vary as they depend on the type of the opcode.
- Consider the following MUL instruction using direct addressing mode:

MUL R1, A

- It multiplies the contents of location A to the contents of register R1 and store the result in R1.

The following sequence of micro-operations occur:

1) The instruction register (IR) contains the MUL instruction. The address part of IR is loaded into the MAR.

2) Then, the memory location corresponding to the address in MAR is read.

3) Then, the data present in location A is transferred to MBR (memory buffer register).

4) The ALU then multiplies the contents of register R1 and MBR.

Hardware design:

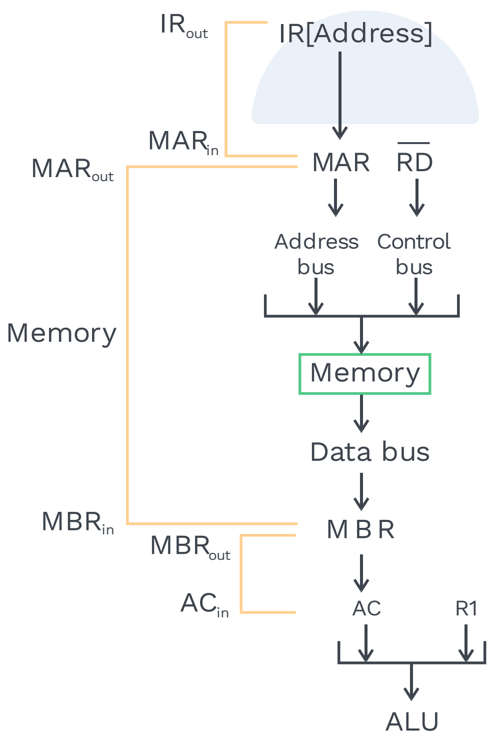


Fig. 3.11 Hardware Design of Execute Cycle

Micro-program:

t_1: MAR ← (IR Address)

t_2: MBR ← Memory

t_3: Accumulator ← MBR

t_4: R1 ← (R1) * (AC)

Write back micro-program:

- After the execution of the instruction, the result might be required to be stored in memory location.

The following sequence of micro-operation occurs:

1) After the execution, the result is stored in the register (accumulator), this result is transferred to the memory buffer register (MBR).

2) IR contains the instruction and the address field of the instruction, which contains the address of the memory location where the result needs to be stored is transferred to MAR.

3) The data present in MBR is stored in the memory location whose address is present in MAR.

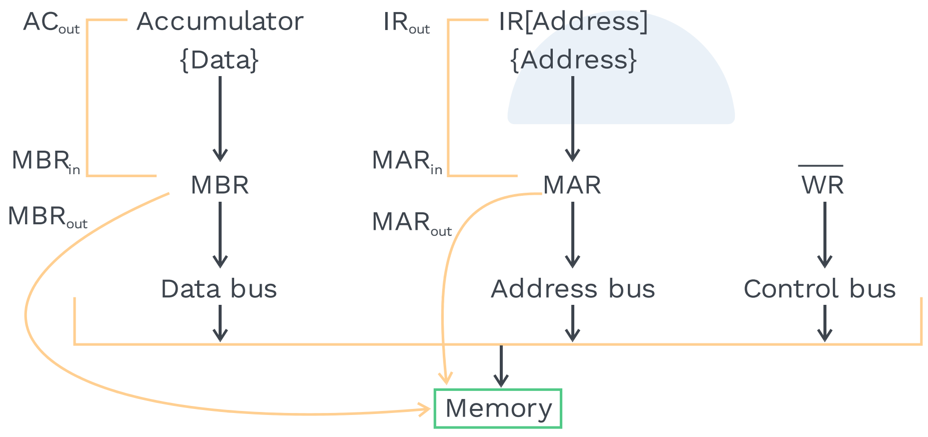


Fig. 3.12 Hardware Design of Write Back

Micro program:

t_1: MBR ← Accumulator

t_2: MAR ← IR [Address]

t_3: Memory ← MBR

The interrupt cycle:

- After the completion of execute cycle, it is checked whether any enabled interrupts have occurred or not.

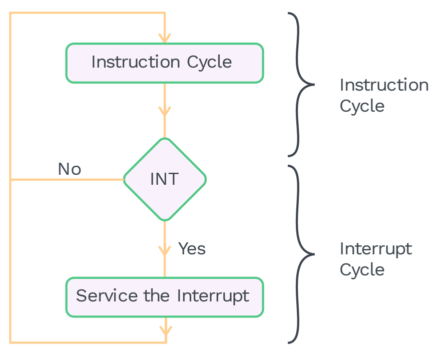


Fig. 3.13 Interrupt Servicing Mechanism

- If interrupts occur, the following sequence of micro-operations takes place:
  1) The next instruction address which is to be executed is present inside the PC. When an interrupt occurs, the address present inside the PC is sent to the memory buffer register(MBR) so that it can be saved for return from the interrupt.
  2) The memory address at which the contents of the PC are to be stored is loaded in MAR.
  3) The starting address of the interrupt processing routine is loaded on the PC.
  4) The last step is to load the contents of MBR, which contains the old value of PC, into the memory whose address is present in the MAR.

Hardware design:

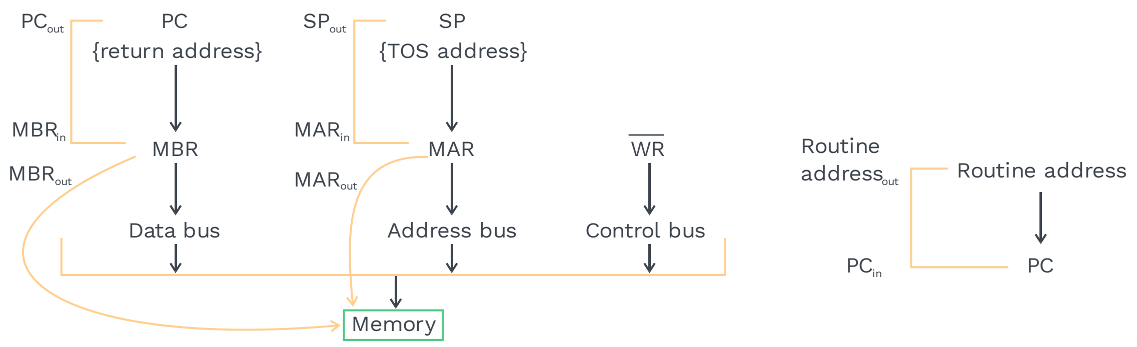


Fig. 3.14 Hardware Design of Interrupt Cycle

Micro-program:

t_1: MBR ← (PC)

t_2: MAR ← Save-address

PC ← Routine address

t_3: Memory ← (MBR)

PRACTICE QUESTIONS

Q1 **Consider a one-word long instruction:**
MOV [2000], @3000
If the processor fetches, decodes, operand fetches, executes and writes back the result of the instruction to the memory. Then the number of times the memory is referred by the processor?

Sol: Fetch: 1 memory reference will be required for fetching the instruction.
Decode: Since the instruction is 1 word long, no memory reference is required during decode.
Operand fetch: Since operand fetch, indirect addressing mode is used, so two memory references will be required.
1 MR for getting the effective address, i.e. address of the operand.
1 MR for getting the operand.
Execute: No memory reference for execute cycle because given instruction is MOV instruction only.
Write Back: One memory reference to write the operand at memory location [2000]
Total memory references = 1 + 2 + 1 = 4

Q2 **For the operand fetch using indirect addressing mode, how many number of cycles are required to bring the operand to the CPU?**

Sol: **Ans. 4**
The microprogram for operand fetch using indirect addressing mode is as follows:

t_1: MAR $\leftarrow$ IR [Address]

t_2: MBR $\leftarrow$ Memory

Thus, in these two cycles effective address, i.e., the address of the operand, is brought to the CPU.

t_3: MAR $\leftarrow$ MBR

t_4: MBR $\leftarrow$ Memory

These two cycles are required to bring the operand from memory to the CPU. So, total of 4 cycles is required.

Q3 **The fetch cycle consists of four micro-operations, and these four micro-operations can be completed in three units of time__________.**
Which among the four micro-operations can be completed in a single time unit in the fetch cycle?

I) MAR ← (PC)

II) MBR ← Memory

III) IR ← (MBR)

IV) PC ← (PC) + Instruction size (I)

a) I and II can be completed in a single time unit.
b) II and IV can be completed in a single time unit.
c) III and IV can be completed in a single time unit.
d) I and IV can be completed in a single time unit.

Sol: **Ans. b), c)**

MBR ← Memory

PC ← (PC) + instruction size (I)

Both these micro-operations can be completed in a single cycle because these two actions, read a word from memory and add instruction size (I) to PC, do not interfere with each other.

IR ← (MBR)

PC ← (PC) + instruction size (I)

Both these micro-operations can be completed in a single cycle because they both do not interfere with each other.
Conflicts occur when reading and writing are done on the same register in single time unit.

Bus architecture:

- Two or more devices in a computer are connected through a common transmission medium known as a bus.
- There are many devices that are connected to the bus, and if any device sends the signal, it can be received by all the other devices connected to the bus.
- Signals that represent binary 1 and binary 0 are transmitted via bus, which contains multiple communication lines.

Previous Years' Question

Consider the following sequence of micro-operations, **(GATE CS 2013)**

MBR ← PC

MAR ← X

PC ← Y

Memory ← MBR

Which one of the following is a possible operation performed by this sequence?

a) Instruction fetch **b)** Operand fetch

c) Conditional branch **d)** Initiation of interrupt service

Sol: d)

System bus:

- The major units of a system, like memory, CPU and I/O are connected by a system bus.
- Typically, the system bus contains the following lines:
  - **a) Data lines:** Data is transferred between various system modules through data lines.
    Data lines altogether are known as the data bus.
  - **b) Address lines:** The address of the data which is present on the data bus is contained by address lines.
    Address lines altogether are known as address bus.
  - **c) Control lines:** The control lines are bearers of control signals that effectively establish the communication of control commands and timing signals among different system components.
- The validity of address and data information is specified by timing signals.
- Control signals describe the operation to be performed like memory write, memory read, I/O write, I/O read etc.

Note:

1) The word length of the CPU can be described based on the width of the data bus.

2) The memory capacity of the system can be described based on the width of the address bus and data bus.

Bus architecture:

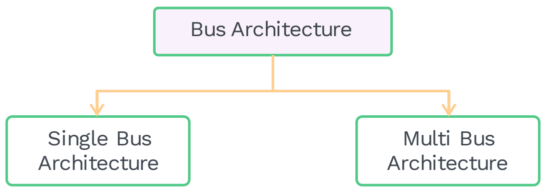


Fig. 3.15 Bus Architecture

1) Single bus architecture:

- To form an operational system, the functional individual units must be connected in some organized way.
- The single bus architecture is the easiest way to interconnect all the functional units.

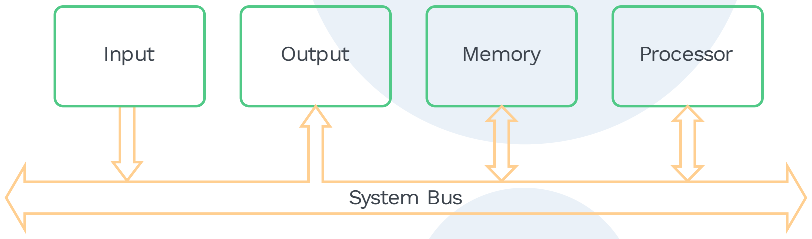


Fig. 3.16 Single-Bus Structure

- Both the instructions and data are transferred through this single bus.

Advantages:

- The single bus architecture is not an expensive design.
- Less number of registers are associated.
- Flexible for attaching new peripheral devices.

Disadvantages:

- Only single transmission can be done by the bus at a given point in time, i.e. only two units can actively use the bus at any given time.
- Less concurrency in operations.
- Requires a greater number of cycles for execution, thus making execution of process slow.

Multi bus architecture:

- The performance of single bus architecture is affected if many devices are connected to the bus.
- With the increase in the count of the components connected to the bus architecture, propagation delay increases. Therefore, devices will take a large amount of time to synchronize the use of the bus.

- Also, when the aggregate data transfer demand becomes so large that it reaches the capacity of the bus and thus bus may become a bottleneck.
- The above problems may not always be overcome by single bus architecture.
- This problem can be solved by using multiple buses, generally laid out in a hierarchy.
- Multiple buses allow the devices to work simultaneously, thus improving the CPU's speed.

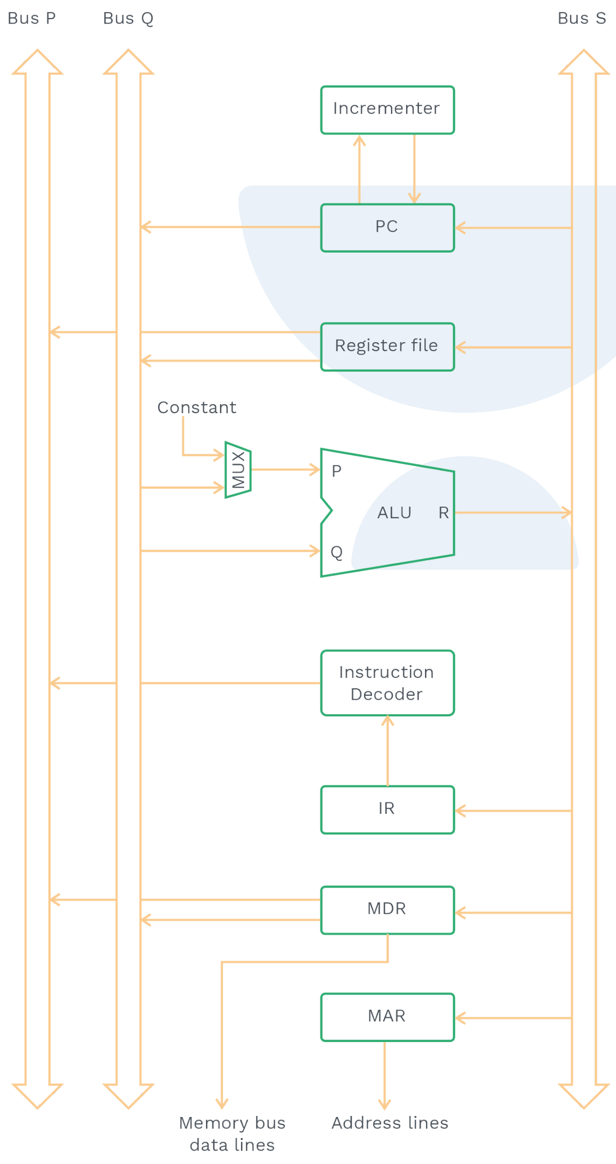


Fig. 3.17 Three Bus Organisation of Data Path

- Buses P and Q are used to transmit the source operands to the P and Q inputs of ALU, where an arithmetic or logic operation can be performed.
- The result is transmitted to the destination over bus S.
- Whenever required, the ALU can forward one of its two input operands, not modified, to bus S.

Advantages:

- The number of cycles required for the execution of the process is less.
- More concurrency in operations.

Disadvantage:

- Multibus architecture is an expensive design.

Control signals:

- Every micro-operation is performed by activating some control signals.
- Control signals are put in directly as binary inputs to the independent logic gates.
- Control signals can be used to activate an ALU function, a data path or other external interfaces.
- The CPU consists of some general purpose and some special purpose registers.
- All the registers are connected to a common data path known as BUS.
- For transferring the information from one register to another, each register has two control signals R_{in} and R_{out}.
- When the content of the bus is loaded into the register R, then R_{in} must be set to 1.
- When the content from the register R is loaded into the bus, then R_{out} must be set to 1.

- Let us consider Fetch Cycle:

Following is the micro-program for the fetch cycle:

t_1: **MAR $\leftarrow$ (PC)**

The control unit turns on the control signals that unlock the gates between the PC and MAR permitting the bits in PC onto MAR.

t_2: **MBR $\leftarrow$ Memory**

PC $\leftarrow$ (PC) + 1

For this micro-operation control unit activates the following control signals:

1) A control signal that unlocks the gates and permits the contents present in MAR onto the address bus.

2) A control signal to read the memory location and transfer the data onto the data bus.

3) A control signal that unlocks the gates and permits the contents on the data bus to be transferred onto the MBR.

4) A control signal that increments the PC value by the instruction length and stores it on PC.

t_3: **IR $\leftarrow$ (MBR)**

A control signal that allows the contents of MBR to be transferred to IR.

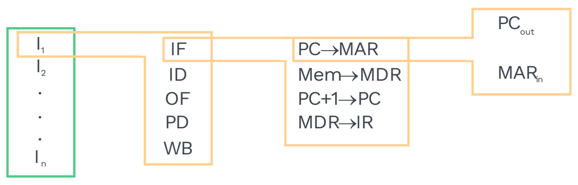


Fig. 3.18 Fetch Cycle

- These control signals can be activated by:
  - **a)** Hardwired control unit design
  - **b)** Micro-programmed control unit design.

Control unit design:

- The implementation of the control unit requires the following information:
  - **1)** Number of instructions implemented in base hardware.
  - **2)** Number of micro-operations required for each instruction.
  - **3)** The control signals which are required for micro-operation.
- The control unit in the CPU is responsible for decoding the instructions and generating the control signals corresponding to the sequence of micro-instructions involved in the instruction.
- There are two approaches to implement control unit:
  - **1)** Hardwired control unit
  - **2)** Micro-programmed control unit

1) Hardwired control unit:

- The hardwired control unit is a sequential state machine to generate the specific fixed sequences of control signals.
- In this design, control signals are expressed in the sum of product expression format.
- Gates ,decoders, flip flops and other digital circuits are used to implement control logic.
- To generate the control signals, fixed logic circuits that directly correspond to Boolean expressions are used.

Advantages:

- Since logic gates and flip flops are used to design control units, so it is the fastest control unit.

Disadvantage:

- It is not flexible because even very small changes in the Boolean expression requires redesign and reconnection of a logic circuit.

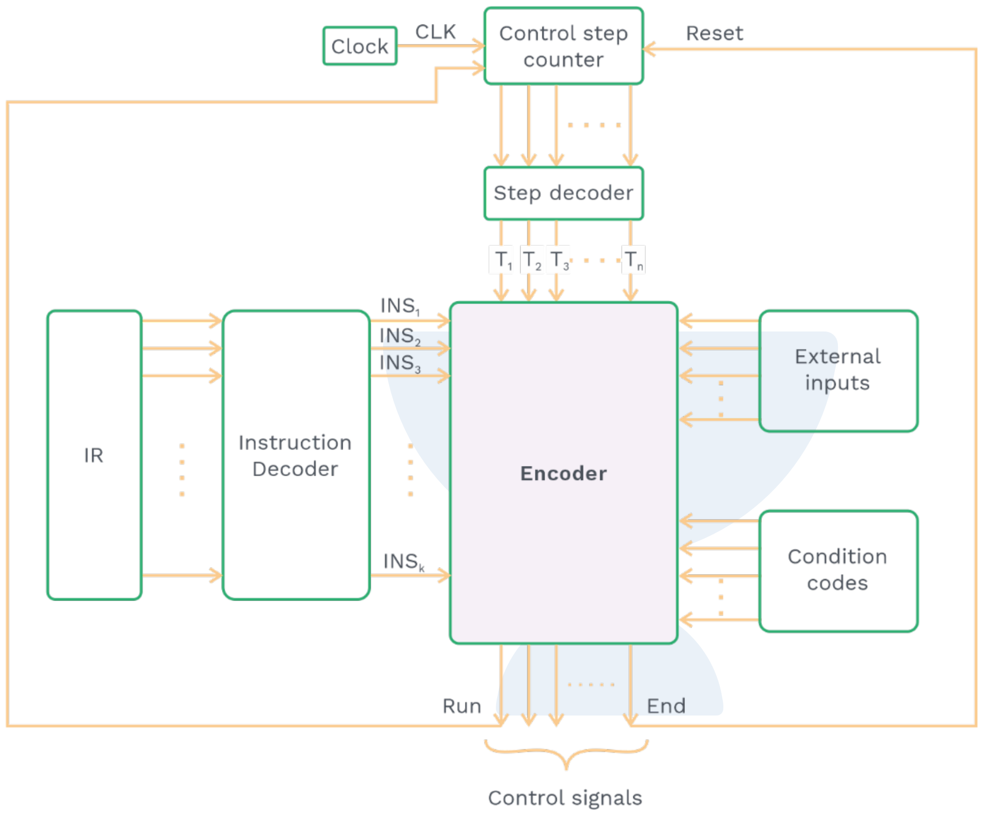


Fig. 3.19 Control Unit Organisation

- The blocks of decoder and encoder are combinational circuits that are used to produce the desired control outputs.
- In control sequence, a different signal line for every step is provided by the step decoder.
- While loading any instruction in IR (Instruction Register), one of the output lines INS_1 through INS_k are set to 1, and all other lines are set to 0.
- The input signals to the encoder block are combined to generate the individual control signals Y_{in}, PC_{out}, Add, End etc.

eg. $Z_{in} = T_1 + T_6 \cdot ADD + T_4 \cdot BR + ...$

- The signal is asserted during time slot T_1 for all instructions, during T_6 for an ADD instruction during T_4 for an unconditional branch instruction and so on.

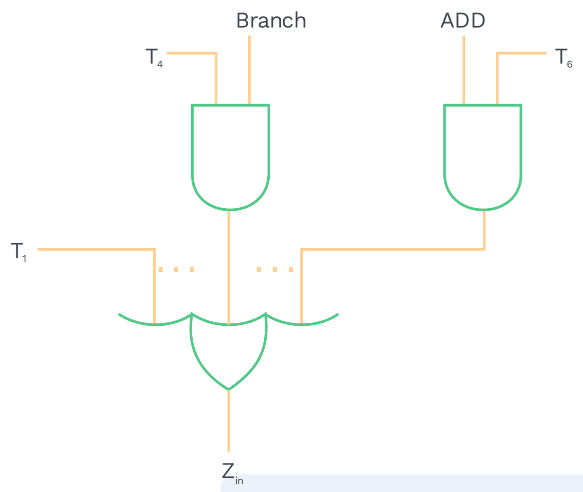


Fig. 3.20 Generation of Z_{in} Control Signal

Note:

Hardwired control unit is used in designing RISC architecture.

PRACTICE QUESTIONS

Q4 **Consider a system in which we have 4 control signals S_0, S_1, S_2, S_3 and 3 different instruction I_1, I_2, I_3 and 4 μ-operations T_1, T_2, T_3, T_4. Following are the control signals which are required for a particular instruction at different μ-operation timings:**

μ-operation/Instruction	**I_1**	**I_2**	**I_3**
T_1	**S_0**	**S_2, S_3**	**S_0, S_1, S_2, S_3**
T_2	**S_0, S_2**	**S_3**	**S_1, S_3**
T_3	**S_3, S_2, S_0**	**S_0, S_3**	**S_1**
T_4	**S_0, S_1, S_2**	**S_3, S_2, S_1**	**S_2**

The expressions for the control signals will be?

Sol: The expression for control signal S_0 will be,

$$S_0 = T_1(I_1 + I_3) + T_2(I_1) + T_3(I_1 + I_2) + T_4(I_1)$$

The expression for control signal S_1 will be,

$$S_1 = T_1(I_3) + T_2(I_3) + T_3(I_3) + T_4(I_1 + I_2)$$

The expression for control signal S_2 will be,

$$S_2 = T_1(I_2 + I_3) + T_2(I_1) + T_3(I_1) + T_4(I_1 + I_2 + I_3)$$

The expression for control signal S_3 will be:

$$S_3 = T_1(I_2 + I_3) + T_2(I_2 + I_3) + T_3(I_1 + I_2) + T_4(I_2)$$

Q5 **Following is the expression for control signal Z, present in a hardware:**

$$\mathbf{Z = T_1(I_1 + I_4) + T_2 + T_3 + T_4\,(I_1 + I_2 + I_3 + I_4)}$$

Given that the system has 5 different instructions I_1, I_2, I_3, I_4, I_5 and each instruction has 4 micro-operations T_1, T_2, T_3, T_4. During the execution of instruction $I_{5,}$ in which micro-operation of the control signal is enabled in the hardware?

a) T_1 and T_2 **b) T_2 and T_3**

c) T_1, T_2, T_3 and T_4 **d) None Sol.**

Sol:

$$Z = T_1(I_1 + I_4) + T_2 + T_3 + T_4\,(I_1 + I_2 + I_3 + I_4)$$

$$\begin{tikzcd}
T_1 \arrow[d] & T_2 \arrow[d] & T_3 \arrow[d] & T_4 \arrow[d] \\
\begin{array}{c} I_1 \\ I_4 \end{array} & \begin{array}{c} I_1 \\ I_2 \\ I_3 \\ I_4 \\ \boxed{I_5} \end{array} & \begin{array}{c} I_1 \\ I_2 \\ I_3 \\ I_4 \\ \boxed{I_5} \end{array} & \begin{array}{c} I_1 \\ I_2 \\ I_3 \\ I_4 \end{array}
\end{tikzcd}$$

So, option ‘b’ is correct.

Previous Years' Question

A hardwired CPU uses 10 control signals S_1 to S_{10}, in various time steps T_1 to T_5, to implement 4 instructions I_1 to I_4 as shown below: **(GATE CS 2005)**

	T_1	T_2	T_3	T_4	T_5
I_1	S_1, S_3, S_5	S_2, S_4, S_6	S_1, S_7	S_{10}	S_3, S_8
I_2	S_1, S_3, S_5	S_8, S_9, S_{10}	S_5, S_6, S_7	S_6	S_{10}
I_3	S_1, S_3, S_5	S_7, S_8, S_{10}	S_2, S_6, S_9	S_{10}	S_1, S_3
I_4	S_1, S_3, S_5	S_2, S_6, S_7	S_5, S_{10}	S_6, S_9	S_{10}

Which of the following pairs of expressions represent the circuit for generating control signals S_5 and S_{10}, respectively?
($(I_j + I_k)T_n$ indicates that the control signal should be generated in time step T_n if the instruction being executed is I_j or I_k).

a) $S_5 = T_1 + I_2 \cdot T_3$ and

$S_{10} = (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

b) $S_5 = T_1 + (I_2 + I_4) \cdot T_3$ and

$S_{10} = (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

c) $S_5 = T_1 + (I_2 + I_4)T_3$ and

$S_{10} = (I_2 + I_3 + I_4) \cdot T_2 + (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

d) $S_5 = T_1 + (I_2 + I_4) \cdot T_3$ and

$S_{10} = (I_2 + I_3) \cdot T_2 + I_4 \cdot T_3 + (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

Sol: d)

Micro-programmed control unit:

- Control Memory is a permanent memory, i.e ROM.
- In this design, micro-programs are used to program the control memory.
- The behaviour of the control unit is depicted by the program present in the control memory.
- Thus, by simply executing that program, the control unit can be implemented.
- The set of micro-instructions is stored in the control memory.

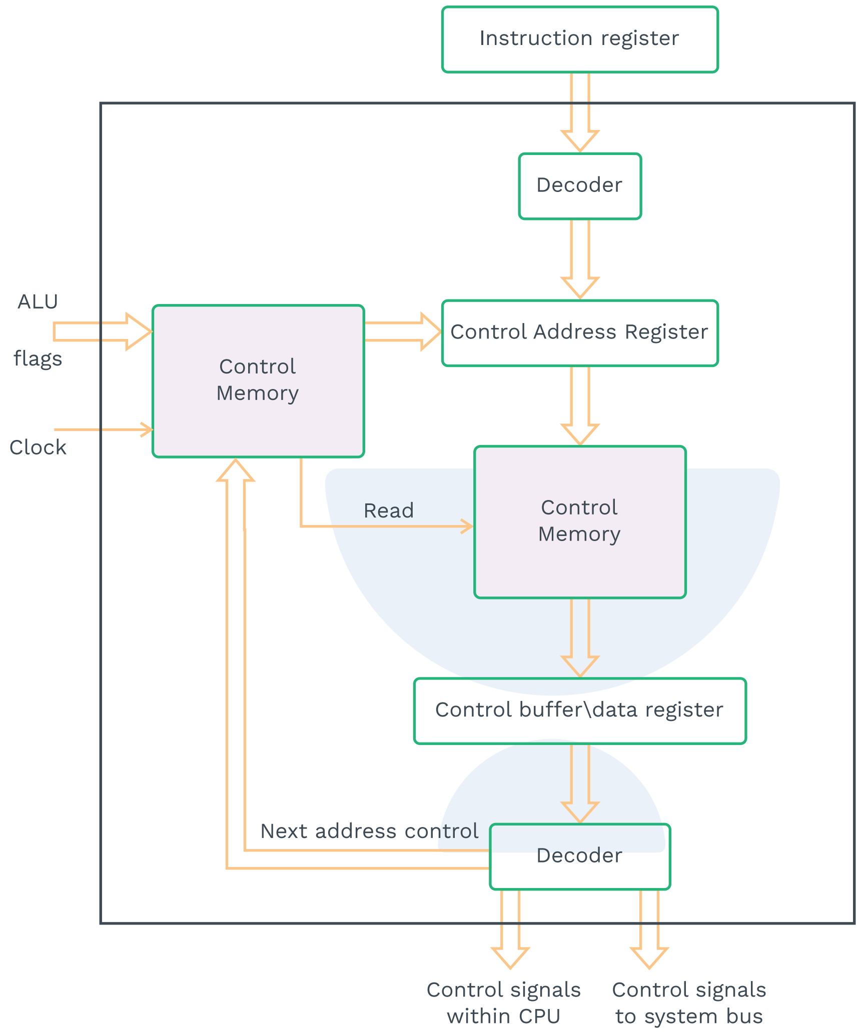


Fig. 3.21 Functioning of Microprogrammed Control Unit

- The address of the next micro instruction to be read is present inside the control address register (CAR).
- After reading micro instruction from the control memory, the micro instruction is transmitted to the buffer register known as a control data register(CDR).
- The sequencing unit loads the control address register and issues a read command.
- Following are the functions of the control unit:
  1) The READ command to the control memory is being issued by the sequencing logic in order to execute an instruction.
  2) The word whose address is given in the control register is read into the control buffer register.

3) Control signals and next address information are generated by the content of the control buffer register.
4) Based on the next address information present inside the CDR and ALU flags, a new address is being loaded by the sequencing the logic unit in the CAR.

- The upper decoder converts the opcode of IR into the control memory address.
- The lower decoder is not used for horizontal micro instructions, but for vertical micro-instructions.
- Control Memory is associated with CAR (Control Memory address register) and CDR (Control Memory data register) registers which are used to hold the control memory address and control memory content, respectively.

Hardware design:

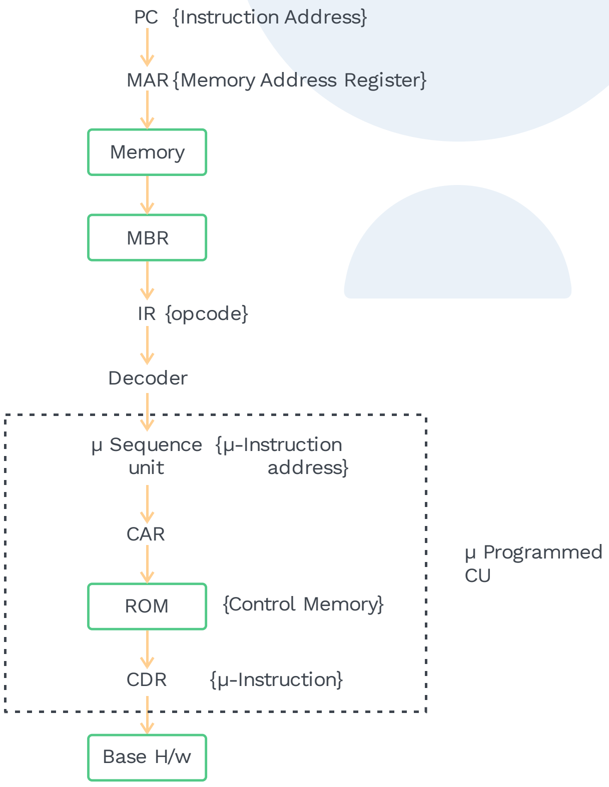


Fig. 3.22 Hardware Design implementing Microprogrammed Control Unit

Advantages:

- The use of micro-programming for implementing the control unit simplifies the design of control unit.

- It's implementation is cheaper and is less error-prone.

Disadvantages:

- Micro-programmed control unit is slower than hard-wired control unit of comparable technology.

Note:

The Micro–programmed control unit is used for designing CISC architectures.

Instruction format:

The μ-Instruction in control memory is stored in the following format:

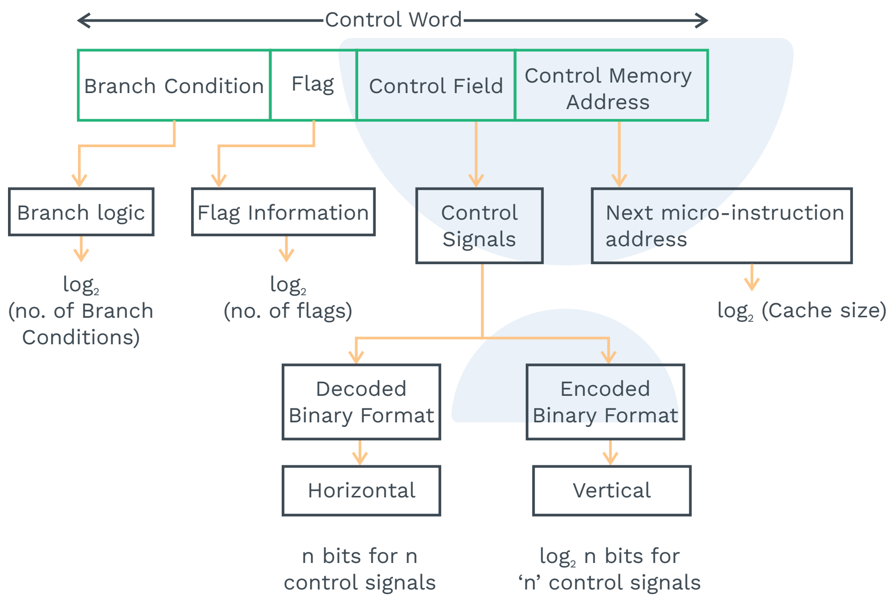


Fig. 3.23 Control Word Design

Note:

Fields (branch condition, flag, control memory address) are used for jumping conditions.

Example: JNZ4004

In this we require three things

1) Condition: Jump if non zero

2) Flag: Zero flag

3) Address: 4004

Micro-instruction sequencing:
The micro-programmed control unit carries out the two important operations listed below as:

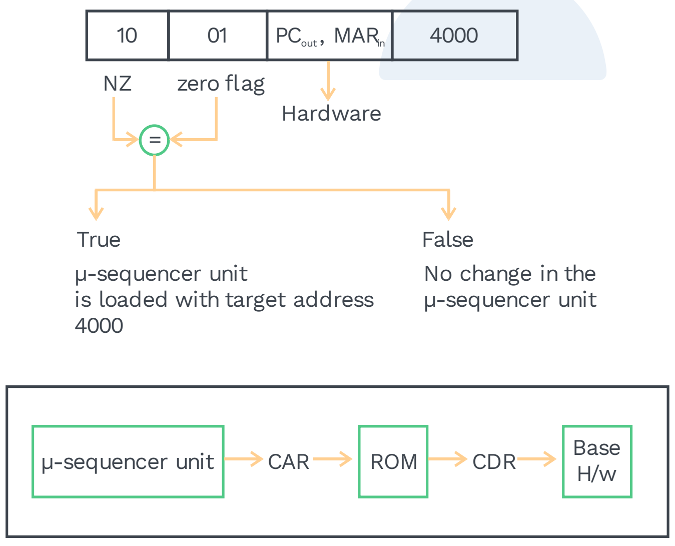
1) **Microinstruction sequencing:** It fetches the subsequent micro-instruction from the control memory.
2) **Microinstruction execution:** It is the procedure of generating the control signals to carry out the execution of the pre-fetched micro-instruction.

Example: Consider the following micro instruction with "Nonzero" branch logic and having target address of 4000.

B.C.	Flag	C.F.	CMA
10	01	PC_{out}, MAR_{in}	4000
NZ	zero flag		

Where {PC_{out}, MAR_{in}} are control signals.
Sol: This micro instruction will be executed as follows:

1) **Micro instruction sequencing:**
Fetch the μ-instruction from the ROM
2) **Micro-instruction execution:**


10
01
PC_out, MAR_in
4000
NZ
zero flag
Hardware
=
True
μ-sequencer unit is loaded with target address 4000
False
No change in the μ-sequencer unit
μ-sequencer unit
CAR
ROM
CDR
Base H/w

μ-Programmed CU

Fig. 3.24 Execution of Micro-Instruction

PRACTICE QUESTIONS

Q6 **Vertical micro-programming, horizontal micro-programming and hardwired control are used for control unit design. Which among them has the highest and lowest operational speed, respectively?**

a) Highest: Hardwired control
b) Highest: Vertical micro-programming
c) Lowest: Vertical micro-programming
d) Lowest: Horizontal micro-programming

Sol: **a), c)**

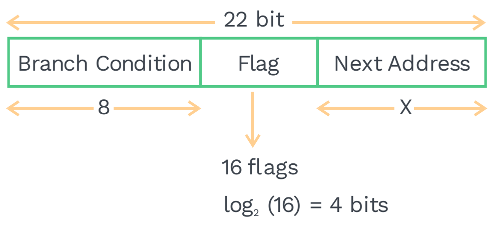
Highest: hardwired control
lowest: vertical micro-programming.

- In hardwired, the control logic is implemented using gates and flipflops. So, it is the fastest.
- In horizontal microprogramming, for each control signal, a separate bit is used.
- In vertical microprogramming, control signals are expressed in the form of encoded binary format.

Thus, a decoder is used.
So, for ‘n’ control signals, $\log_2 n$ bits are used, so it has the slowest operational speed.

Q7 **Consider a control memory unit in which a 22-bit micro-instruction is stored.**
The micro-instruction has three fields, the “next address” field of size ‘X’ bits, the “branch condition” field of 8 bits and the hardware containing 16 flags.
Then what is the size of the control memory?

Sol: **Ans.** 2816 bytes
μ-Instruction format:



Number of bits for next address field (X)

= 22 – (8 + 4)

= 10 bits

Size of control Memory = 2^{10} * 22 bits = (2^{10}) * 22/8 = 2816 bytes.

Horizontal micro-programmed control unit:

- Control signals in horizontal micro programmed control units are expressed in the decoded binary format.
- In this control unit design, each control signal is associated with 1 bit.
- Hence, 'n' number of control signals will require decoding of 'n' bits.

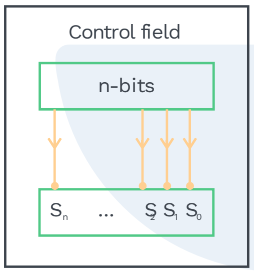


Fig. 3.25 Hardware Design of Microprogrammed Control Unit

- This micro-programmed control unit is fast, as all the control signals are enabled altogether.

Vertical micro-programmed control unit:

- Control signals in vertical micro programmed control units are expressed in encoded binary format.
- Hence, 'n' number of control signals will require encoding of '$\log_2 n$' bits.

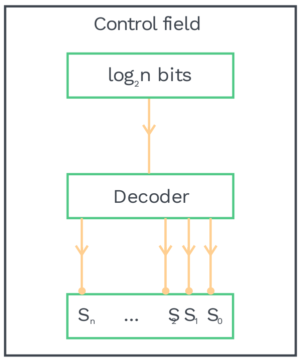


Fig. 3.26 Hardware Design of Vertical Microprogrammed Control Unit

- This micro-programmed control unit is slow since any one control signal can be enabled altogether.

Micro-instruction design using horizontal micro-programming:

- If for instruction 'I', T is one of the micro instructions having 4 control signals (S_0, S_1, S_2, S_3).
- Suppose the following control signals are enabled.
  T: {S_0, S_1, S_3}

μ-instruction design:

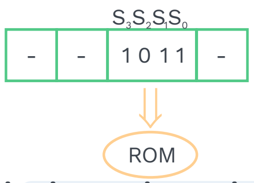


Fig. 3.27 Microinstruction Design Example

Execution:

- Fetch T from the control memory (ROM)

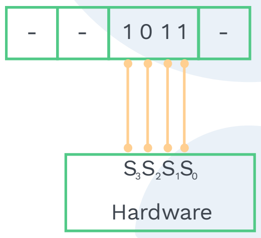


Fig. 3.28 Hardware Execution

Since no branch condition is given, so,
μ-sequencer unit $\rightarrow$ CAR (control address register)

Micro-instruction design using vertical microprogramming:

- If for instruction I, T is one of the micro-instruction having 4 control signals (S_0, S_1, S_2, S_3).
- If the following control signals are enabled

  T {S_0, S_1, S_3}

- μ-Instruction format:

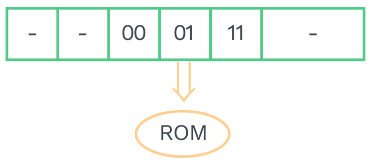


Fig. 3.29 Vertical Microinstruction Design Example

Execution:

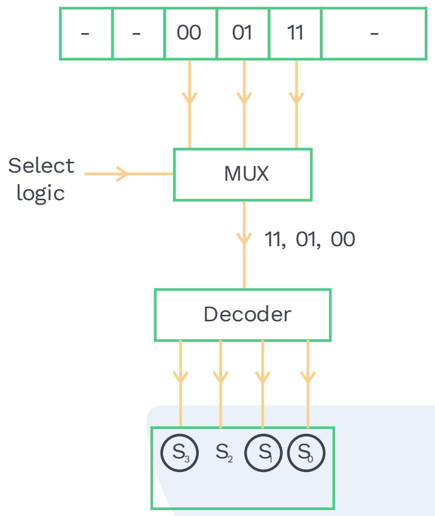


Fig. 3.30 Hardware Execution

Note:

1) Control field in the horizontal micro programming control unit is fixed.
2) Control field in the vertical micro programming control unit is variable, i.e. it depends on the number of function code.

	Horizontal micro-programming		Vertical micro-programming
1)	In horizontal micro-programming, the control signals are expressed in decoded binary format.	**1)**	In vertical micro-programming, the control signals are expressed in encoded binary format.
2)	High degree of parallelism is allowed, as in this none or more than one signal can be enabled at any instance of time.	**2)**	It allows a low degree of parallelism, as in this control unit none or only one signal can be enabled at any instance of time.
3)	It is faster as each control signal is denoted by a bit. So, an external decoder is not needed.	**3)**	It is slower as 'n' control signals require $\log_2 n$ bits. So, a external decoder is required to generate control signals.

Horizontal micro-programming		Vertical micro-programming	
4)	It is less flexible	**4)**	It is more flexible.
5)	The control unit, using horizontal micro-programming is very complex and expensive to design.	**5)**	The control unit, using vertical micro-programming is comparatively easy and cheaper to design.
6)	It supports longer control word.	**6)**	It supports shorter control word.

Table 3.1 Comparison between Horizontal and Vertical Microprogramming

PRACTICE QUESTIONS

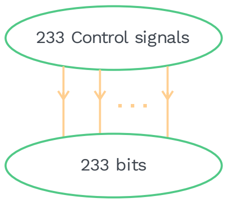
Q8 **Consider a hypothetical control unit, which has a total of 233 control signals. How many bits are required in control memory word for horizontal micro-programming and vertical microprogramming, respectively?**

a) 233, 233 **b) 233, 1** **c) 8, 8** **d) 233, 8**

Sol: **Ans. d)**

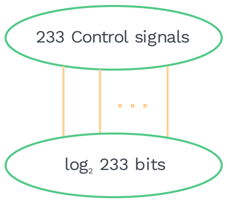
Horizontal micro-programming:

In horizontal micro-programming, for each control signal one bit is used.



Vertical micro-programming:

In vertical micro-programming, for n control signals, $\log_2 n$ control bits are required because in this design, control signals are expressed in an encoded binary format.



Therefore, $\log_2 233 = 8$ bits are required.

Q9 **Consider a micro-programmed control unit design which supports 7 groups of mutually exclusive control signals.**

Groups	**Gr1**	**Gr2**	**Gr3**	**Gr4**	**Gr5**	**Gr6**	**Gr7**
Control signals	**2**	**10**	**4**	**1**	**18**	**23**	**6**

How many more control bits are required using horizontal micro-programming over vertical micro-programming?

Sol: **Ans. 43**

For vertical micro programming:

Micro-Instruction format:

-	-	Gr1	Gr2	Gr3	Gr4	Gr5	Gr6	Gr7	-
		$\log_2 2$	$\log_2 10$	$\log_2 4$	$\log_2 1$	$\log_2 18$	$\log_2 23$	$\log_2 6$	

Total control bits required = 1 + 4 + 2 + 1 + 5 + 5 + 3.

= 21 bits

[For horizontal micro-programming]

Micro instruction format:

-	-	Gr1	Gr2	Gr3	Gr4	Gr5	Gr6	Gr7	-
		2	10	4	1	18	23	6	

Total bits required = 2 + 10 + 4 +1 + 18 + 23 + 6

= 64 bits

So, horizontal micro programming requires 64-21 = 43, more bits.

Previous Years' Question

Consider a CPU where all the instructions require 7 clock cycles to complete execution. There are 140 instructions in the instruction set. It is found that 125 control signals are needed to be generated by the control unit. While designing the horizontal micro-programmed control unit, a single address field format is used for branch control logic. What is the minimum size of the control word and control address register?

a) 125, 7 **b) 125, 10** **c) 135, 9** **d) 135, 10**

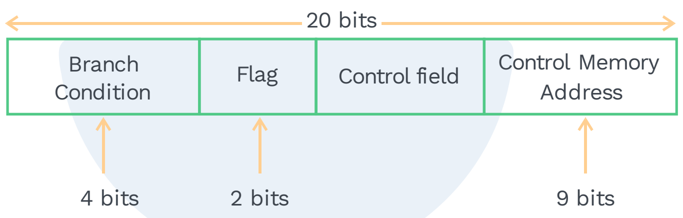
Sol: d) **(GATE CS 2008 (IT))**

Q10 **Consider a hypothetical control unit, which supports 512 bytes of the control word memory. If the control word is 20 bits long and hardware contains 4 flags and supports 16 branch conditions. What is the maximum number of control signals that can be generated at a time using horizontal micro-programming and vertical micro-programming, respectively?**

a) 1, 1 **b) 32, 32** **c) 32, 1** **d) 1, 32**

Sol: **Ans. c)**

μ-instruction design:



Control word memory = 512 bytes

So, control memory address field requires = 9 bits

Number of flags = 4

So, number of flag bits required = 2

Number of branch conditions = 16

So, number of branch condition bits required = 4

Number of bits required for control field = 20 – (4 + 2 + 9) = 5

Therefore, the number of control signals that can be generated = 2^5 = 32

In horizontal micro-programming, for each control signal, 1 bit is there,

So, all control signals can be activated at a time.

$\therefore$ Maximum degree of parallelism for horizontal microprogramming = 32

In vertical micro programming only 1 control signal can be activated at a time.

$\therefore$ Maximum degree of parallelism for vertical micro-programming = 1

$\therefore$ Option c) is correct.

Chapter summary

- **Introduction:** To process the instructions, the CPU uses three components:
- Registers
- ALU (arithmetic and logic unit)
- Control unit (CU)
- **Control unit functions:** Two basic tasks performed by the control unit are:
- Sequencing
- Execution
- **Micro-operations:** Micro-operations are functional or atomic operations of a processor that completes its execution in 1 cycle.
- **Micro-program:** Sequence of micro-instructions used to execute a task.

It includes the following cycles:

1) Fetch cycle
- Instruction Fetch
- Operand fetch

2) Execute cycle
3) Write back
4) Interrupt cycle

- **Bus architecture:** To form an operational system, functional individual units must be converted in some organized way:
  1) Single bus architecture
  2) Multi bus architecture
- **Control unit Design**
  1) Hard wired control unit
  2) Micro programmed control unit
- **Hard wired control unit:** Control logic is implemented with logic gates, flip-flops and other digital circuits
- **Micro-programmed control unit:** In this micro-program is used to program control memory.
  1) Horizontal micro-programmed control unit.
  2) Vertical micro-programmed control unit.