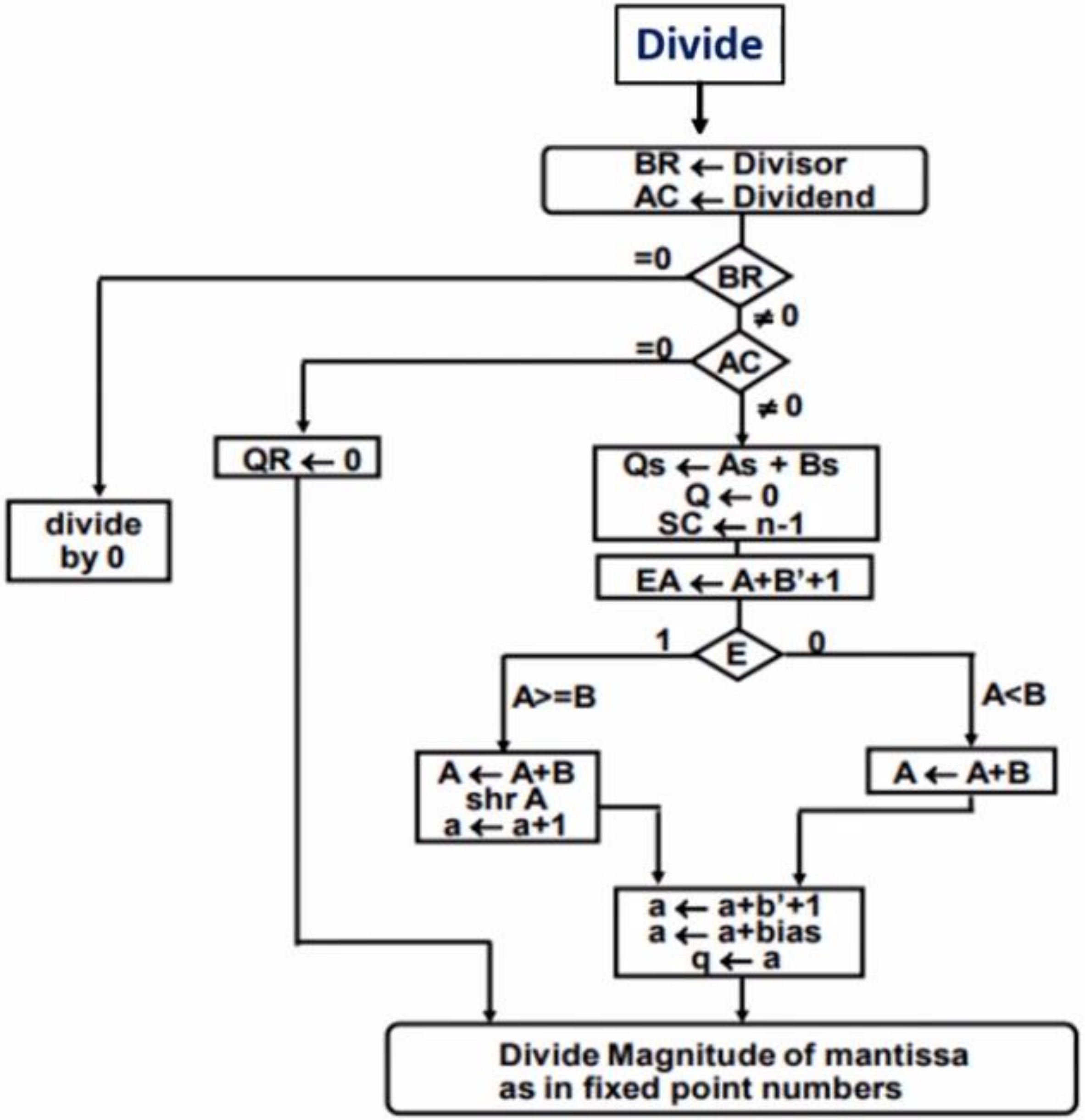
Floating Point Division



Flowchart for Floating Point Division

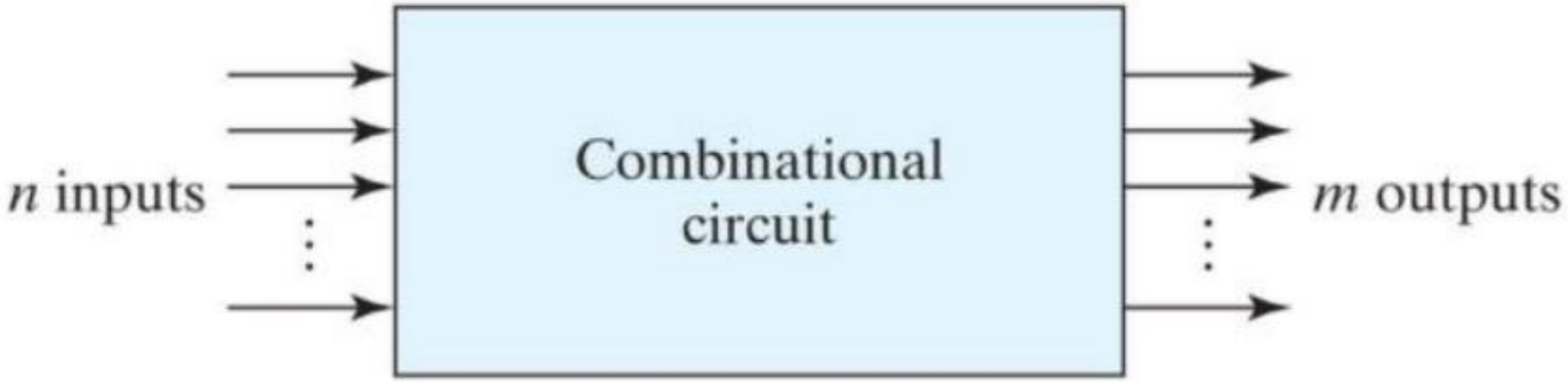
Floating-point division does not require an alignment of mantissas, **division** can be formed by **dividing two mantissas** and **subtracting the exponents**

The **algorithm** can be **divided into five steps:**

- Check for zeros
- Initialize registers and evaluate the sign
- Align the dividend
- Subtract the exponents
- Divide the mantissas

Combinational Circuits

- **Combinational circuits** are defined as the time independent circuits which do not depends upon previous inputs to generate any output are termed as combinational circuits



Combinational Circuits

Examples of Combinational Circuits

- Multiplexer
- Demultiplexer
- Encoder
- Decoder
- HalfAdder
- FullAdder

Combinational Circuits

1.In this output depends only upon present input.
2.Speed is fast.
3.It is designed easy.
4.There is no feedback between input and output.
5.This is time independent.
6.Elementary building blocks: Logic gates
7.Used for arithmetic as well as boolean operations.
8.Combinational circuits don’t have capability to store any state.
9.These circuits do not have any memory element.
10.It is easy to use and handle.

Sequential Circuits

- **Sequential circuits** are those which are dependent on clock cycles and depends on present as well as past inputs to generate any output.

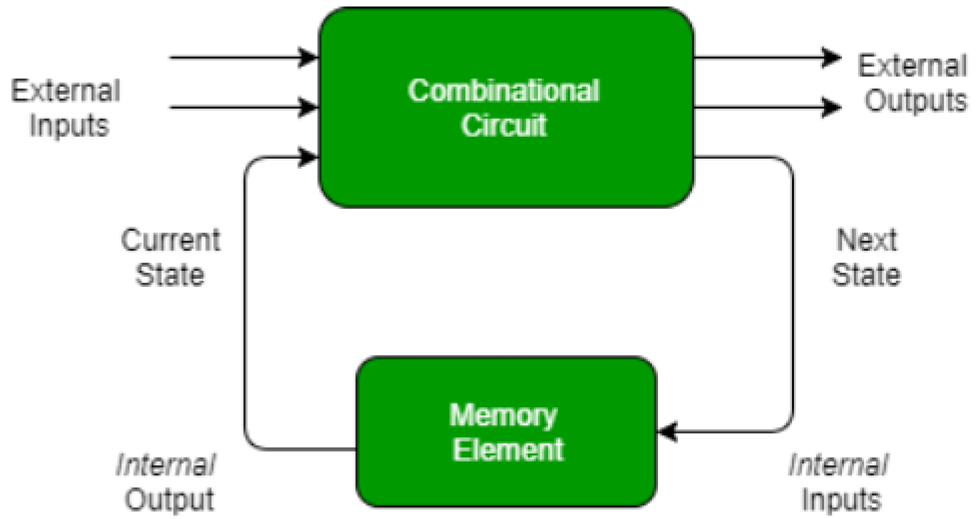


Figure: Sequential Circuit

Sequential Circuits

Examples of Sequential Circuits

- Flip-Flops
- Registers
- Counters

Sequential Circuits

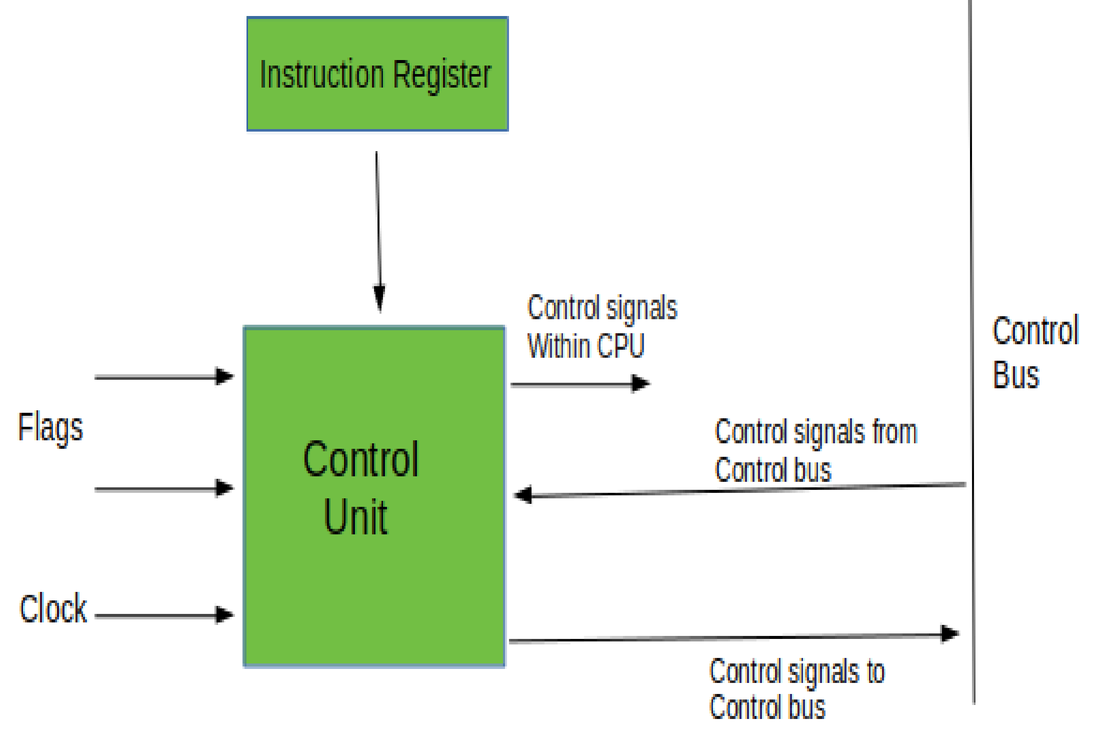
1. In this output depends upon present as well as past input.
2. Speed is slow.
3. It is designed tough as compared to combinational circuits.
4. There exists a feedback path between input and output.
5. This is time dependent.
6. Elementary building blocks: Flip-flops
7. Mainly used for storing data.
8. Sequential circuits have capability to store any state or to retain earlier state.
9. These circuits have memory element.
10. It is not easy to use and handle.

CH: CONTROL UNIT

- The Central Processing Unit is the most important part of a computer. It is responsible for regulating the various operations which are undertaken by a computer.
- A Central Processing Unit or the CPU has three main parts which are the Arithmetic Logic Unit (ALU), the control unit (CU), and the Memory Unit. The control unit is an important component of the CPU.
- It directly controls the functions of the memory unit, the ALU, and the input and output devices. It handles all the signals of the processor and is also regarded as the brain of the processor. Control Unit takes input from the status registers and the instruction registers.

What is the Control Unit?

- A control unit is an integral part of the central processing unit of the computer. The main function of this is to instruct the memory, the arithmetic logic unit, and the input and output devices on how they should process a particular instruction.
- It provides timing and control signals to the various other units of the **CPU**. All the important resources of the computer system are managed by this unit. It also enables data transfer between the various devices and the **CPU**. This is also included as a part of Von Neumann architecture.


Instruction Register
Control Unit
Flags
Clock
Control signals Within CPU
Control signals from Control bus
Control signals to Control bus
Control Bus

Block Diagram of the Control Unit

- It is the responsibility of the Control Unit to tell the computer’s memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor.
- It fetches internal instructions of the programs from the main memory to the processor instruction register, and based on this register contents, the control unit generates a control signal that supervises the execution of these instructions.
- A control unit works by receiving input information to which it converts into control signals, which are then sent to the central processor.
- The computer’s processor then tells the attached hardware what operations to perform.
- The functions that a control unit performs are dependent on the type of CPU because the architecture of CPU varies from manufacturer to manufacturer.

- Input components include flags, clock, and control signals.
- **Clock:** The clock enables the control unit to maintain time. One microinstruction (or a set of microinstructions) can be performed in one clock pulse.
- **Flags:** Flags are required to determine the outcome of the preceding ALU operation and the exact status of the processor.
- **Control Signals from the control bus:** The control bus generates control signals such as acknowledgment and interrupts signals. These signals are sent to the control unit.
- Instruction Register is used to storing the instruction which is currently being executed.

.

- Output Components
- The outputs include control signals inside the processor and the control signals to the control bus.
- **Control Signals inside the processor:** These signals are used to transfer the data from one register to another. Some control signals are also responsible for activating certain **ALU functions**.
- **Control Signals to the control bus:** This includes control signals to the memory and control signals to the input/output devices.

- Functions of the Control Unit
- This unit enables the sequential flow of data between the different parts of the Central Processing Unit.
- It can interpret the instructions and perform the necessary actions.
- This handles multiple operations like fetching, decoding, and execution of instructions.
- It controls the flow of data inside the process.
- This also controls the execution units like ALU, registers, and data buffers which are present in the CPU.

As for example, consider the instruction : “Add contents of memory location NUM to the contents of register R1 and store the result in register R1.” For simplicity, assume that the address NUM is given explicitly in the address field of the instruction .That is, in this instruction, direct addressing mode is used.

Execution of this instruction requires the following action :

1. Fetch instruction
2. Fetch first operand (Contents of memory location pointed at by the address field of the instruction)
3. Perform addition
4. Load the result into R1.

Steps	Actions
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

Steps	Actions
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

In Step1:

The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a read request to memory.

To perform this task first of all the contents of PC have to be brought to internal bus and then it is loaded to MAR.To perform this task control circuit has to generate the PC_{out} signal and MAR_{in}signal.

After issuing the read signal, CPU has to wait for some time to get the MFC signal. During that time PC is updated by 1 through the use of the ALU. This is accomplished by setting one of the inputs to the ALU (Register Y) to 0 and the other input is available in bus which is current value of PC.

At the same time, the carry-in to the ALU is set to 1 and an add operation is specified.

Steps	**Actions**
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

In Step 2:

The updated value is moved from register Z back into the PC. Step 2 is initiated immediately after issuing the memory Read request without waiting for completion of memory function. This is possible, because step 2 does not use the memory bus and its execution does not depend on the memory read operation.

In Step 3:

Step3 has been delayed until the MFC is received. Once MFC is received, the word fetched from the memory is transferred to IR (Instruction Register), Because it is an instruction. Step 1 through 3 constitute the instruction fetch phase of the control sequence.

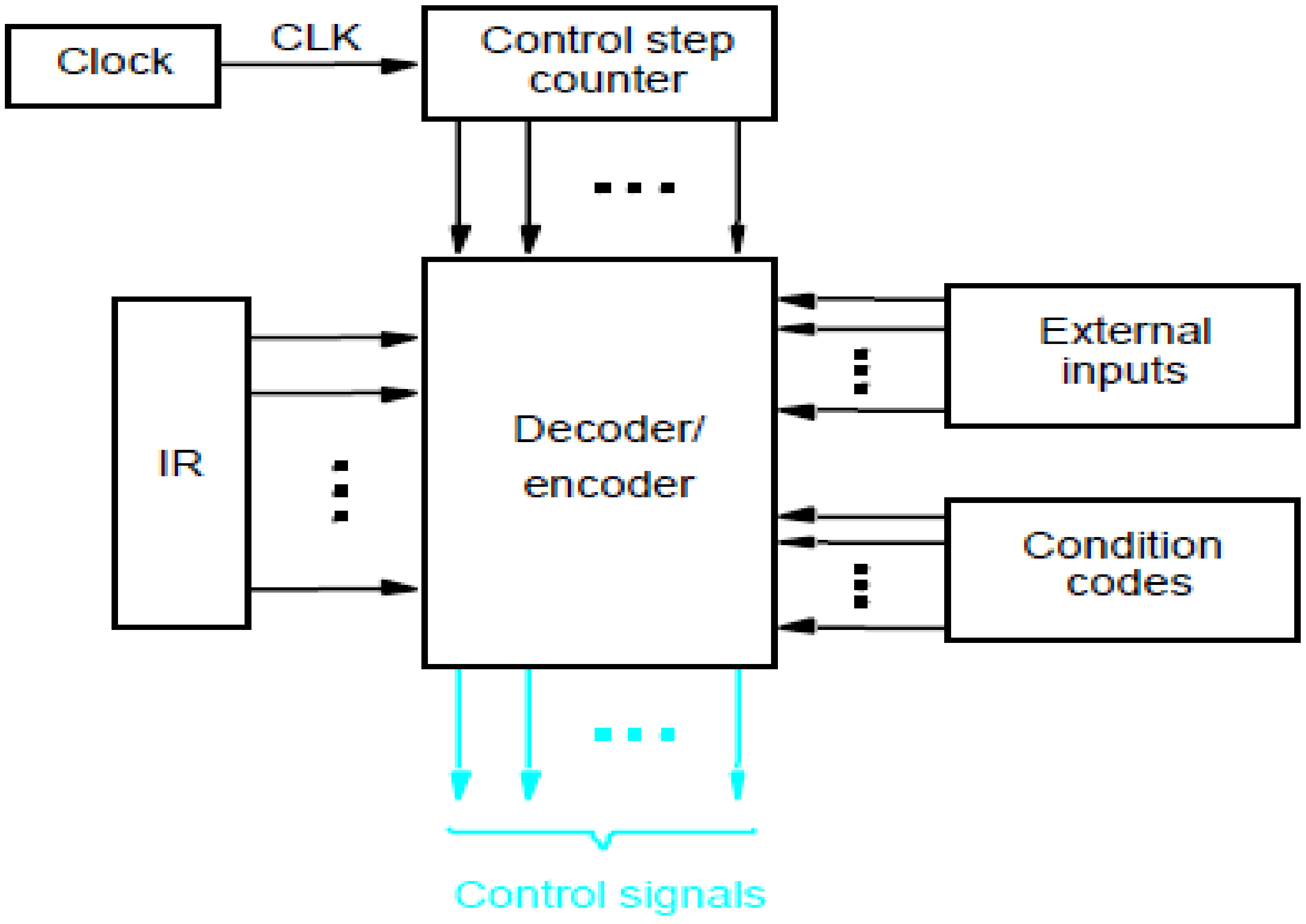
The instruction fetch portion is same for all instructions. Next step inwards, instruction execution phase takes place.

Steps	Actions
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

As soon as the IR is loaded with instruction, the instruction decoding circuits interprets its contents. This enables the control circuitry to choose the appropriate signals for the remainder of the control sequence, step 4 to 8, which we referred to as the execution phase. To design the control sequence of execution phase, it is needed to have the knowledge of the internal structure and instruction format of the PU. Secondly , the length of instruction phase is different for different instruction.

- **Types of Control Unit –**
  There are two types of control units: Hardwired control unit and Microprogrammable control unit.

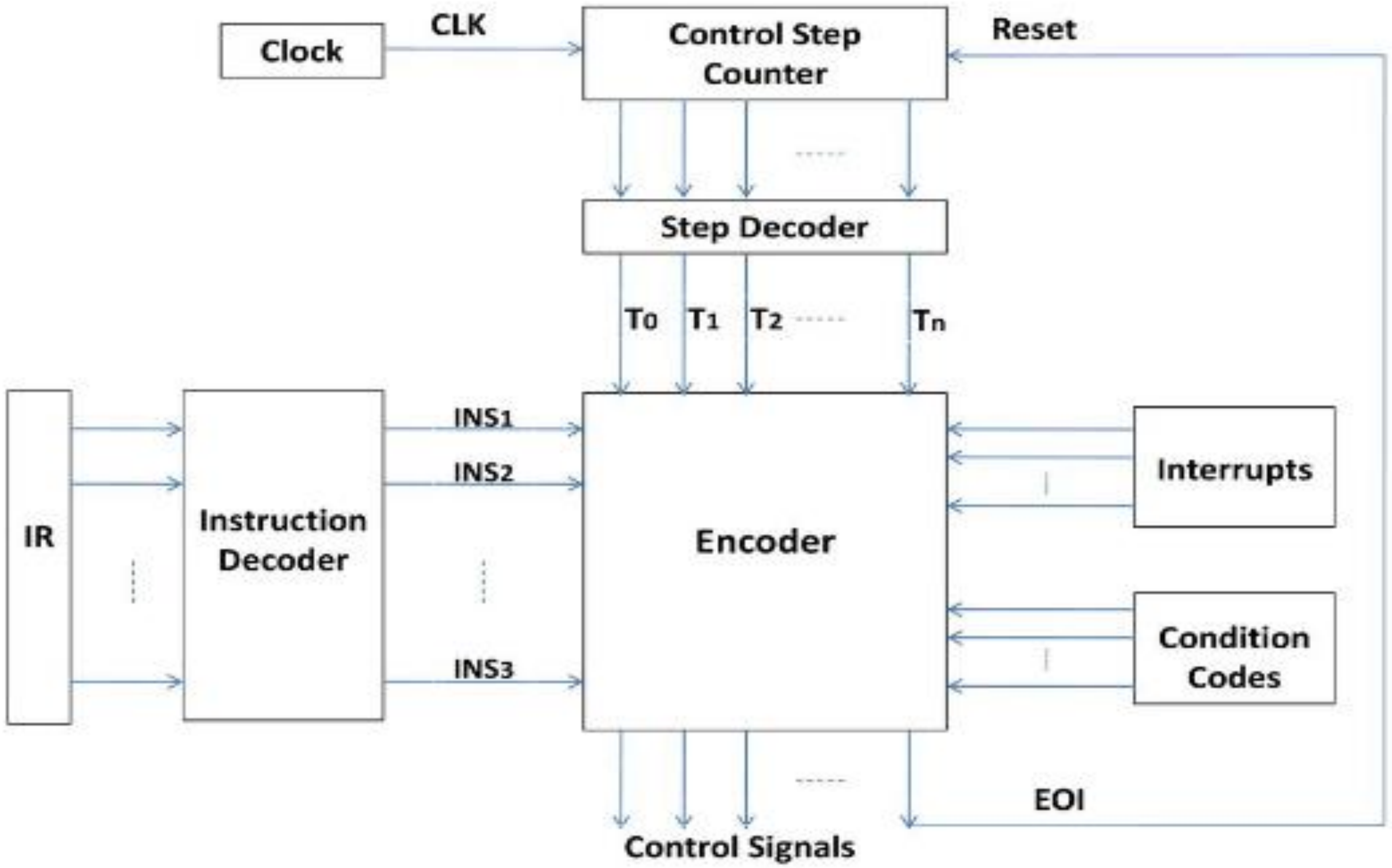
- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: hardwired control and microprogrammed control
- The hardwired system can operate at high speed, but with little flexibility.



Control Unit Organization

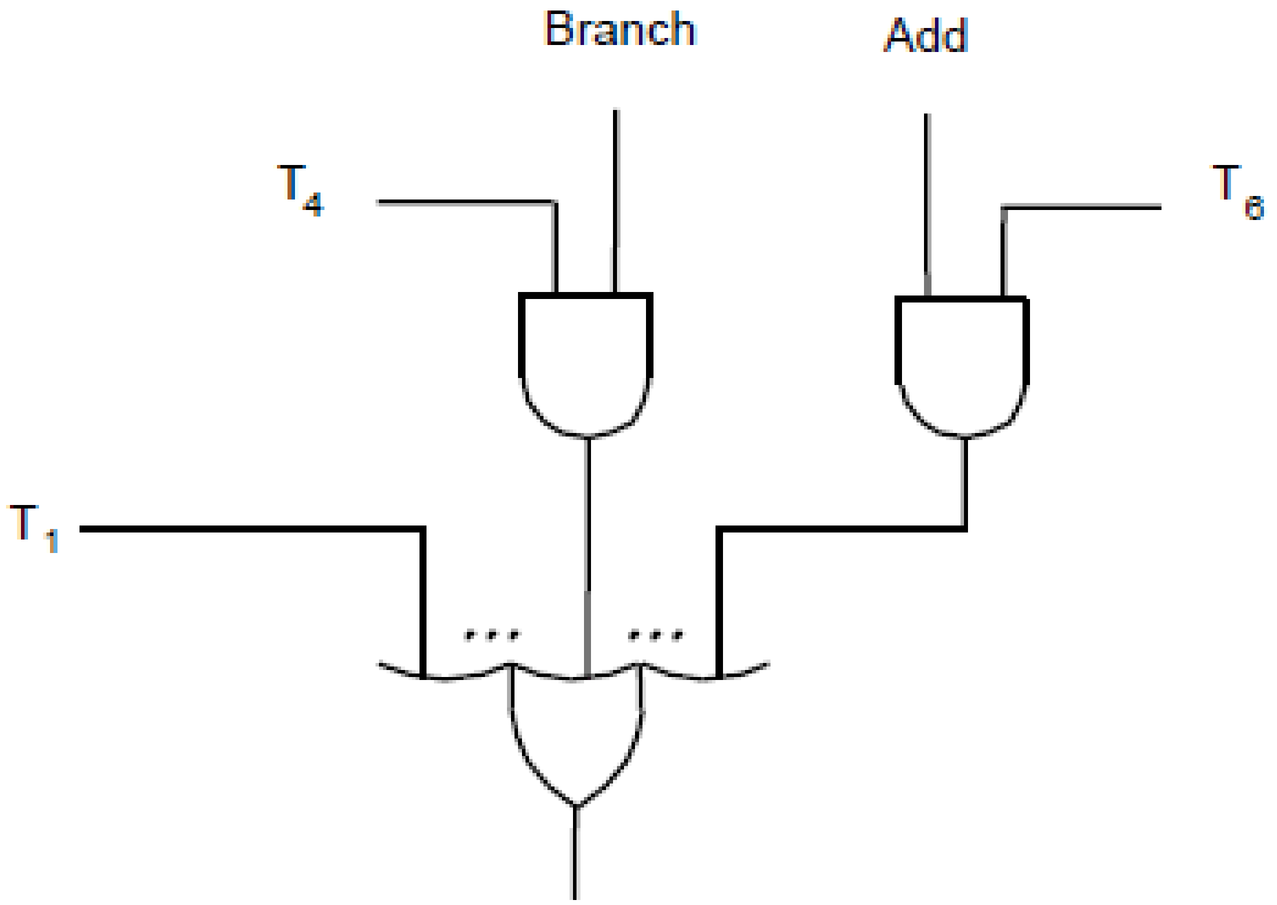
- The processor keeps track of information about the results of various operations. This is accomplished by recording the required information in individual bits, called **Condition Code Flags.**
  • These flags are grouped together in a special processor-register called the condition code register (or statue register).
- • Four commonly used flags are:

1) N (negative) set to 1 if the result is negative, otherwise cleared to 0.
2) Z (zero) set to 1 if the result is 0; otherwise, cleared to 0.
3) V (overflow) set to 1 if arithmetic overflow occurs; otherwise, cleared to 0.
4) C (carry) set to 1 if a carry-out results from the operation; otherwise cleared to 0.



Separation of Encoding and decoding functions

Generating Zin

- Zin = T1 + T6 • ADD + T4 • BR + ...
- 

- **Advantages of Hardwired Control Unit :**

1. Because of the use of combinational circuits to generate signals, Hardwired Control Unit is fast.
2. It depends on number of gates, how much delay can occur in generation of control signals.
3. It can be optimized to produce the fast mode of operation.
4. Faster than micro- programmed control unit.

- **Disadvantages of Hardwired Control Unit :**

1. The complexity of the design increases as we require more control signals to be generated (need of more encoders & decoders)
2. Modifications in the control signals are very difficult because it requires rearranging of wires in the hardware circuit.
3. Adding a new feature is difficult & complex.
4. Difficult to test & correct mistakes in the original design.
5. It is Expensive.

Microprogrammed Control Unit

- In microprogrammed control, in which control signals are generated by program similar to machine language program

- Consider the following control sequence to execute the instruction ADD (R3), R1.

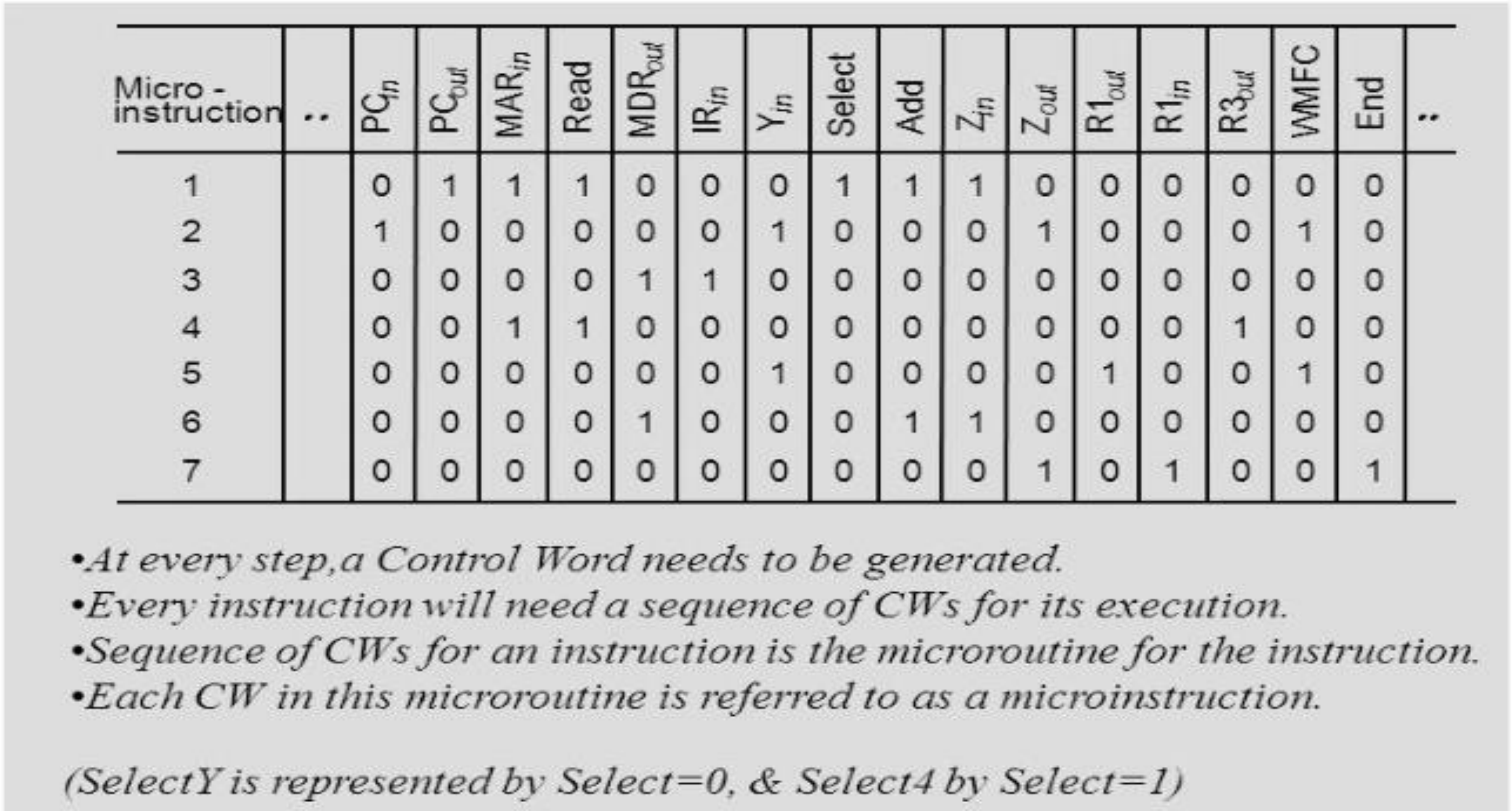
Control Word (CW) is a word whose individual bits represent various control signals.

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4,Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

At every step, some control signals are asserted (=1) and all others are 0.

Control Signals:
PC_{out}
PC_{in}
MAR_{in}
Read
MDR_{out}
IR_{in}
Y_{in}
SelectY
Select4
Add
Z_{in}
Z_{out}
$R1_{out}$
$R1_{in}$
$R3_{out}$
WMFC
End ......

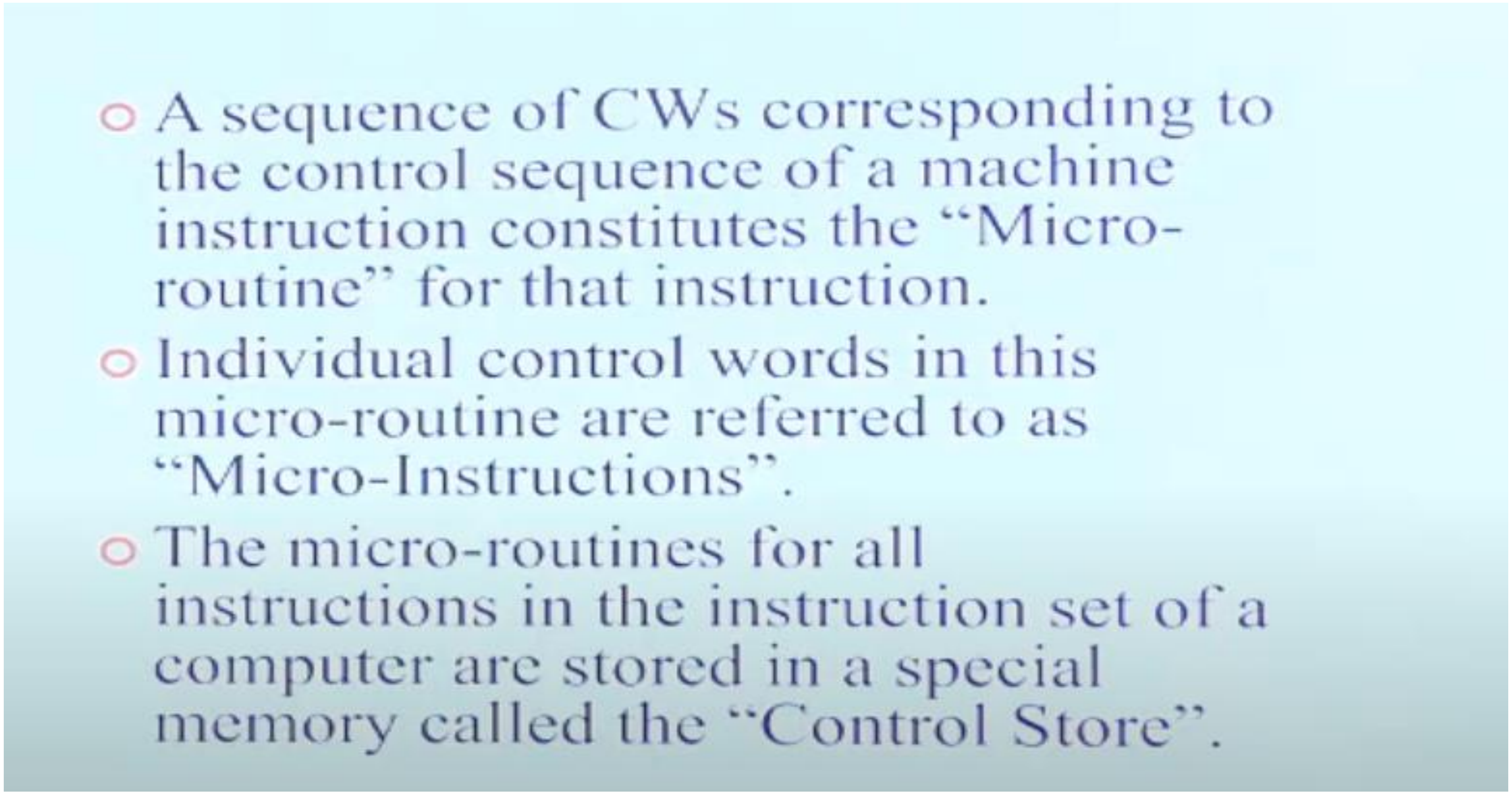
Microprogrammed control (contd..)



Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

- *At every step,a Control Word needs to be generated.*
- *Every instruction will need a sequence of CWs for its execution.*
- *Sequence of CWs for an instruction is the microroutine for the instruction.*
- *Each CW in this microroutine is referred to as a microinstruction.*

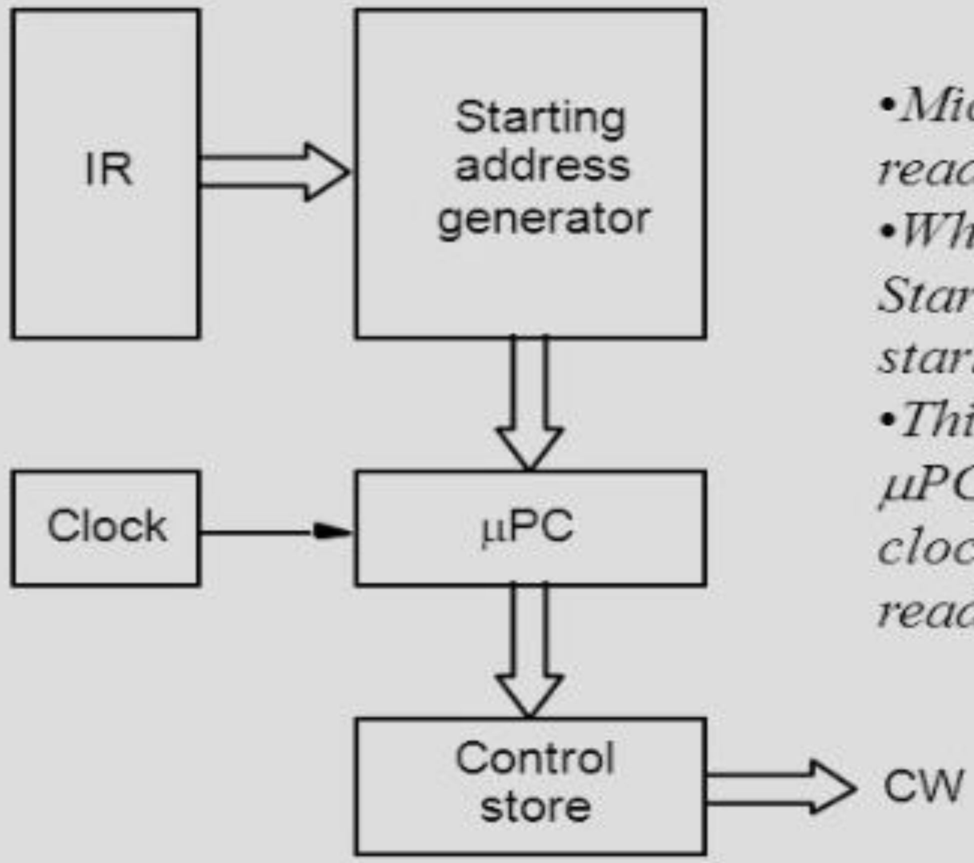
(SelectY is represented by Select=0, & Select4 by Select=1)



- A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the "Micro-routine" for that instruction.
- Individual control words in this micro-routine are referred to as "Micro-Instructions".
- The micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the "Control Store".

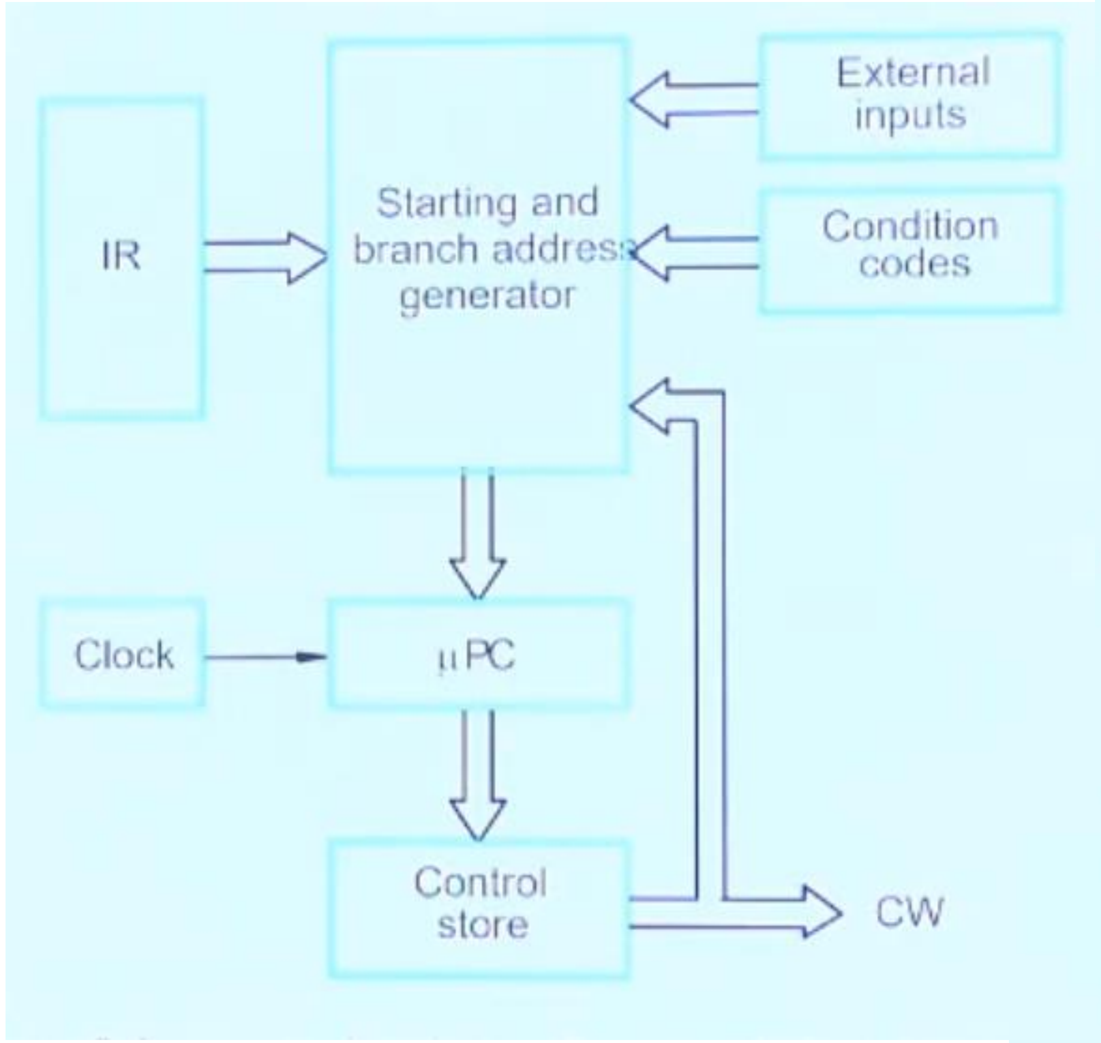
Microprogrammed control (contd..)

Basic organization of a microprogrammed control unit.



•Microprogram counter (µPC) is used to read CWs from control store sequentially.
•When a new instruction is loaded into IR, Starting address generator generates the starting address of the microroutine.
•This address is loaded into the µPC. µPC is automatically incremented by the clock, so successive microinstructions are read from the control store.

- The previous organization cannot handle the situation when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.
- Use conditional branch microinstruction.



Organization of control unit to allow conditional branching in the micro program

- The "Starting and Branch Address Generator" loads a new address into the μPC .
- In this control unit, the μPC is incremented every time a new microinstruction is fetched from the micro-program memory, except in the following situations:
- 1. When a new instruction is loaded into the IR, the μPC is loaded with starting address of the micro-routine for that instruction.
- 2. When a branch instruction is encountered and the branch condition is satisfied, the μPC is loaded with the branch target address.
- 3. When an End microinstruction is encountered, the μPC is loaded with the address of the first CW in the micro-routine for the instruction fetch cycle.

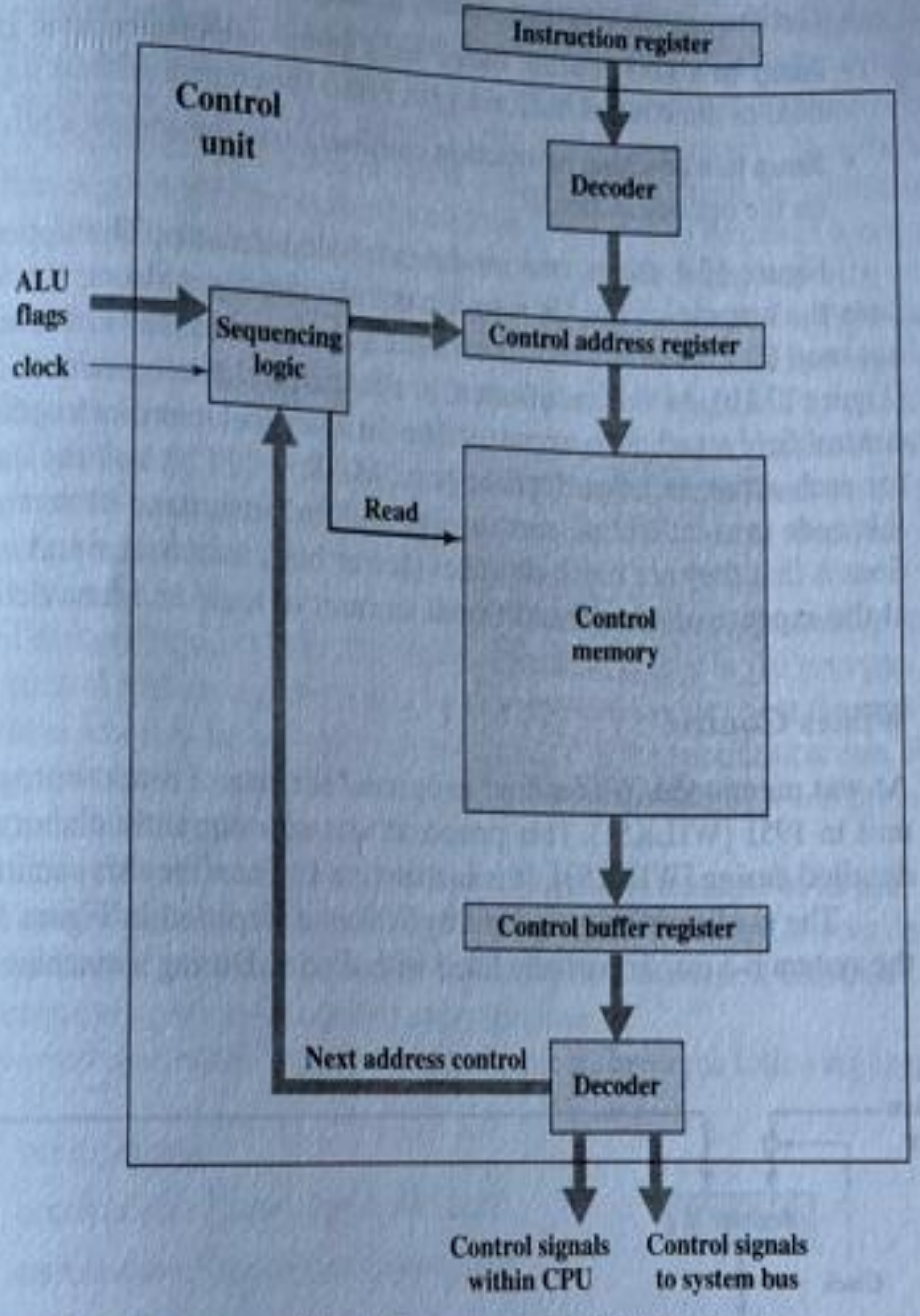


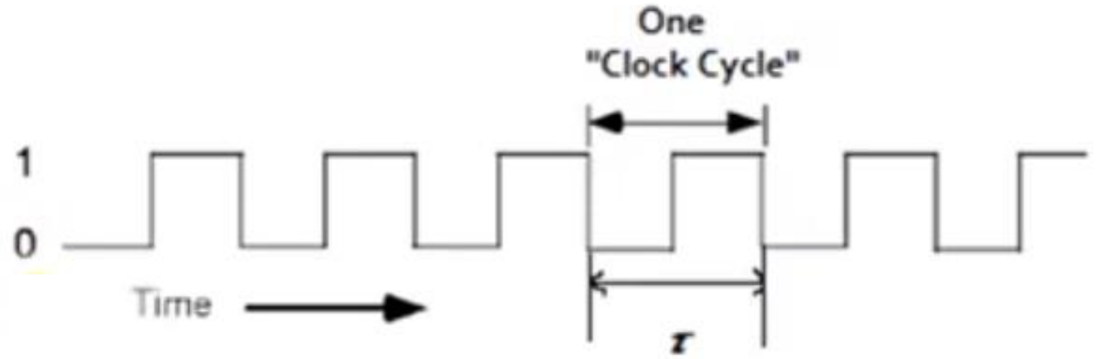
Figure 17.4 Functioning of Microprogrammed Control Unit

1. To execute an instruction, the sequencing logic unit issues a READ command to the control memory.
2. The word whose address is specified in the control address register is read into the control buffer register.
3. The content of the control buffer register generates control signals and next-address information for the sequencing logic unit.
4. The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffer register and the ALU flags.

- **Turn around Time** : It is *simplest measure of program performance*. It is the time which includes disk and memory access, input and output activities, compilation time, OS overhead and CPU time.
- Other important attributes to calculate CPU Performance are:
  1. **Clock rate and CPI**
  2. **Execution Time**
  3. **MIPS Rate**
  4. **Throughput Rate (Performance)**

1. Clock rate and CPI

- *CPU Clock:* Computers are constructed in such a way that events in hardware are synchronized using a **clock**



- *Cycle Time/ clock time/clock period (τ):*

  CPU is driven by a **clock** of constant **cycle time τ (in ns)**

- *Clock rate/ clock frequency (f):* is defined by the **inverse of cycle time**

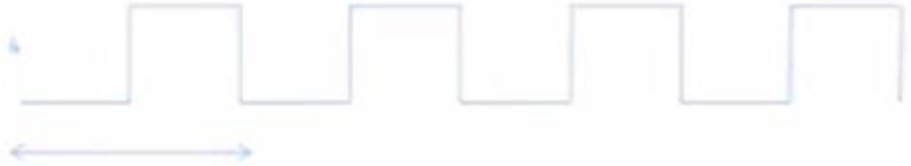
$$\boxed{\text{Clock rate } f = 1/\tau}$$ **in Hz (= cycle/second)**

Also, Cycle time $\tau = 1/f$

Example

1. $f = 1$ GHz $= 10^9$ Hz

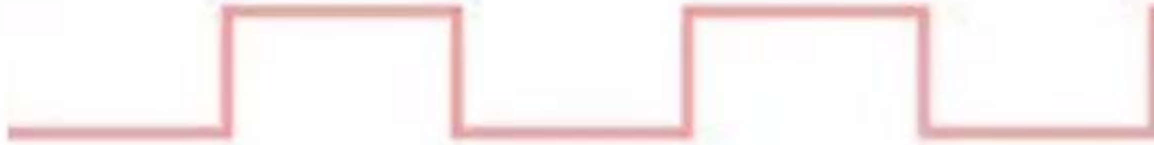
   $\tau = \frac{1}{f} = \frac{1}{10^9}$ SPC = 1ns



2. $f = 500$ MHz

   $= 500 * 10^6$ Hz

   $\tau = \frac{1}{f} = \frac{1}{500 * 10^6 \text{ Hz}}$ SPC = 2ns



Clock rate and CPI …

- ***Instruction count (Ic)***: *Instruction count (Ic) is the **size of a program**, which is the no. of machine instructions to be executed in a program. Sometimes called instruction path length*
- *A machine instruction is a sequence of micro-operation. A micro-operation is an elementary hardware operation that can be carried out in one clock cycle.*
- *Thus a single machine instruction may take one or more CPU cycles to complete.*
  - *We can characterize an instruction by* **Cycles Per Instruction (CPI)**
- *CPI* **:** Different machine instructions may require different number of clock cycles to execute. Therefore ***CPI (Cycles Per Instruction)*** becomes an important parameter.
  - ***CPI measures the time needed to execute each instruction i.e. number of clock cycles per instruction for a program***

$$\boxed{CPI = \frac{\text{CPU Clock cycles for a program}}{\text{Instruction Count}} = \frac{C}{Ic}}$$

Clock rate and CPI …

- ***Average (or Effective) CPI of a program:***
  - Average (or Effective) CPI of all instructions executed in the program on a given processor
  - Different instructions can have different CPIs

2. Execution Time (CPU Time)

- *Execution Time (CPU Time): It is the time needed to execute the program*

  Thus the *Execution Time / CPU time (T in seconds / program) is* given by:

$$\boxed{T = Ic * CPI * \tau} \quad \text{----(1)}$$

Where *Ic* = *Total no. of instructions executed*

CPI = *average number of Cycles per instruction (CPI) for a program*

τ = *Clock cycle time (Clock period)*

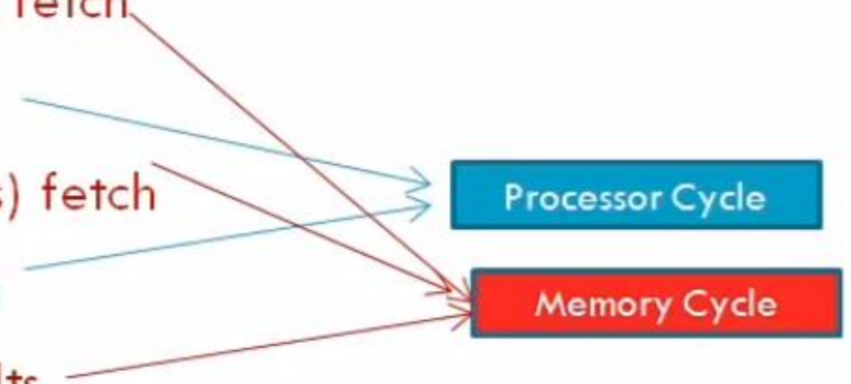
T = Execution Time per program in seconds

- *The **units** for CPU Execution time are:*

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

Execution Time (CPU Time)…

- The execution of instruction (i.e. Instruction cycle) requires some steps:
  - instruction fetch
  - **decode**
  - operand(s) fetch
  - **execution**
  - store results



- In instruction cycle, only decode and execution phases are carried out in the CPU (Processor cycle). The remaining three operations may require memory access (memory cycle).
- Therefore, ***CPI = Instruction cycle = processor cycles + memory cycles.***

  Usually, memory cycle is k *times* the processor cycle τ

  CPI = Instruction cycle = p + m * k

Execution Time (CPU Time)....

Therefore, CPU time T is given by:

$$\boxed{T = I_c * (p + m * k) * \tau} \qquad \text{----(2)}$$

Where

I_c = instruction count

p = number of processor cycles

m = number of memory references

k = ratio between memory and processor cycle

τ = processor cycle time

T = CPU Time (Execution Time)

System Attributes

- The 5 performance factors (I_c, p, m, k, τ) are influenced by 4 system attributes:

System Attributes	Ic	p	m	k	τ
Instruction set architecture.	X	X			
Compiler technology.	X	X	X		
CPU implementation & control		X			X
Cache & memory hierarchy				X	X

1. **Instruction-set architecture (ISA)** affects Ic (program length), p (processor cycle)
2. **Compiler technology** *affects* Ic, p *and* m (memory reference count)
3. **CPU implementation and control** *determines the* total processor time $= p * \tau$
4. **Cache and memory hierarchy** *affects* memory access time$= k * \tau$

3. MIPS Rate

- Let C be the ***Total number of clock cycles*** *needed to execute a program.*

  Therefore, *CPU time (T) = Total number of clock cycles * clock cycles*

  $$T = C * \tau = C / f$$

  Furthermore, as we know $CPI = C / Ic$ So, $\boxed{C = CPI * Ic}$

  and $T = CPI * Ic * \tau$

  Therefore,

  $$\boxed{T = \frac{Ic * CPI}{f}} \qquad \text{----(3)}$$

- **The processor speed is often measured in terms of MIPS (Millions Instructions Per Second).** It is simply called **MIPS Rate** of a given processor.

  $$\boxed{MIPS\ Rate = \frac{Ic}{T * 10^6} = \frac{f}{CPI * 10^6} = \frac{f * Ic}{C * 10^6}} \qquad \text{----(4)}$$

  Using equation (4)

  $$\boxed{T = \frac{Ic}{MIPS * 10^6}}$$

4. Throughput rate (Performance)

- **Throughput (Ws) or Performance**: The number of programs executed per unit time is called system throughput (in programs /second)

$$W_s = \frac{1}{T} = \frac{1}{\text{Execution time}}$$

$$\boxed{W_s = \frac{f}{I_c * CPI} = \frac{MIPS * 10^6}{I_c}}$$

Instruction types and CPI

- Consider a program executing on a processor with **n** types or classes of instructions (like load, store, ALU, branch, etc.)

We *know* ***CPI = CPU Clock cycles for a program (C) / Instruction Count (Ic)***

Ici = number of instructions of type *i* executed

CPIi = cycles per instruction for type *i*

So,

For CPU design:

$$\boxed{CPU\ clock\ cycles = CPI * I_c = \sum_{i=1}^{n} (CPI_i * I_{ci})}$$

The overall CPI is given by:

$$\boxed{Average\ (or\ Effective)\ CPI\ \ \sum_{i=1}^{n} (CPI_i * I_{ci}) = \frac{\sum_{i=1}^{n} (CPI_i * I_{ci})}{I_c} = \sum_{i=1}^{n} \left(CPI_i * \frac{I_{ci}}{I_c}\right)}$$

Example 1

- For the multi-cycle MIPS Processor, there are five types of instructions:

  Load (5 cycles)

  Store (4 cycles)

  R-type (4 cycles)

  Branch (3 cycles)

  Jump (3 cycles)

  If a program has:

  50% load instructions

  25% store instructions

  15% R-type instructions

  8% branch instructions

  2% jump instructions

  then, the CPI (or Average CPI) is:

$$CPI = \frac{\sum_{i=1}^{n} (CPI_i * I_{ci})}{I_c}$$

$$\text{CPI} = \frac{5 \times 50 + 4 \times 25 + 4 \times 15 + 3 \times 8 + 3 \times 2}{100} = 4.4$$

Example 2

- An instruction set has three instruction classes A, B, C:

Instruction class (Eg: ALU, Branch, etc)	CPI
A	1
B	2
C	3

- Two code sequences have the following instruction counts (*Ic*):

Code Sequence	Instruction counts *(Ic)* for instruction class A	B	C
1	2	1	2
2	4	1	1

- ***CPU cycles for sequence 1*** = *CPI * Ic* = 2 x 1 + 1 x 2 + 2 x 3 = 10 cycles

  CPI for sequence 1 = *clock cycles / Ic* = 10 /5 = 2
- ***CPU cycles for sequence 2*** = *CPI * Ic* = 4 x 1 + 1 x 2 + 1 x 3 = 9 cycles

  CPI for sequence 2 = *clock cycles / Ic* = 9 / 6 = 1.5

Factors in the Performance Equation

- CPU Execution time
  - Running the program (Seconds for the program)
- Clock Cycle Time
  - Published as part of the documentation for a computer (Seconds per clock cycle)
- Instruction Count
  - Depends on the architecture (Instructions executed for the program)
- CPI
  - Varies by application, as well as among implementations with the same Instruction set (Average number of clock cycles per instruction)

SPEC

- Programs used to measure performance
  - Specialized benchmarks for particular classes of applications
- Standard Performance Evaluation Corporation (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- Elapsed time to execute a selection of programs
  - Negligible I/O, so focuses on CPU performance

Other Performance Metrics

- **Power consumption**
  - Especially in the embedded market where battery life is important
    - For power-limited applications, the most important metric is energy efficiency
  - Clock Rate and Power both are co-related
  - So, both increased rapidly for decades and then flattened off
  - And now there is a practical power limit for cooling commodity microprocessors

Other Performance Metrics

- **Amdahl's Law**
  - A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used

In other words, it is a formula used to find the maximum improvement possible by just improving a particular part of a system. It is often used in *parallel computing* to predict the theoretical speedup when using multiple processors.