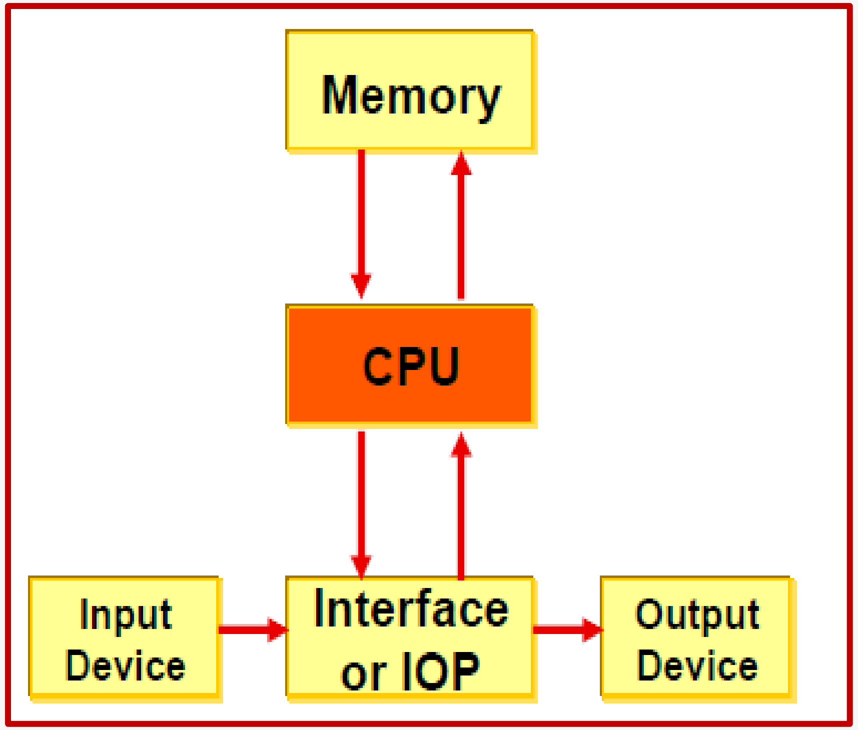
Computer Architecture

Agenda

1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

Introduction to digital computer architecture

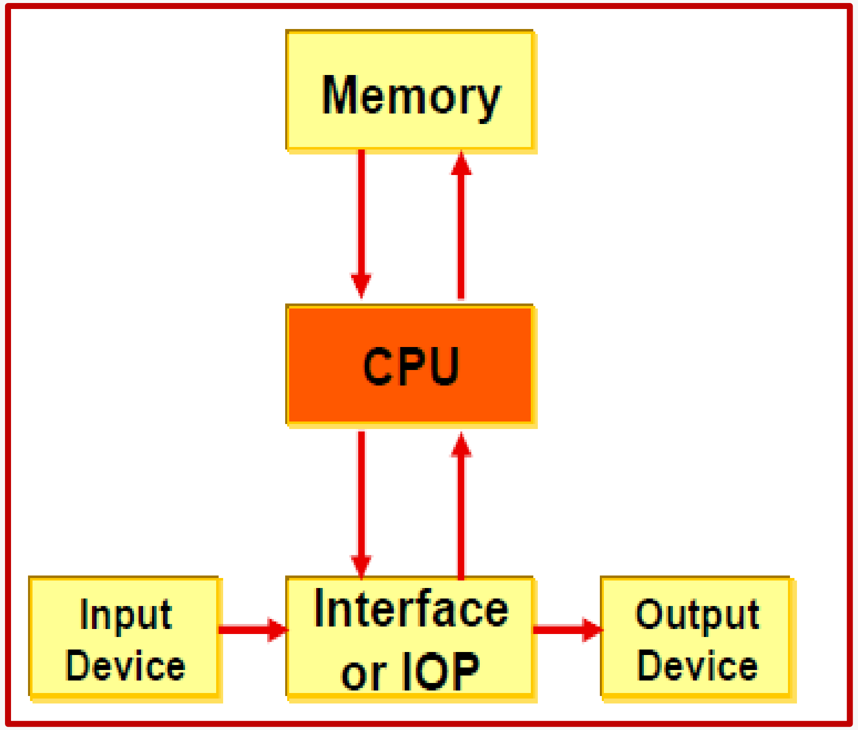
- Any digital computer should be consisted of the shown basic components.
- **CPU** is used to read instructions, understand them and execute different micro operations .
- **Memory** is used to hold program instructions and program data.



Introduction to digital computer architecture

- **Interface** unit is used to make the interaction between the CPU and the surrounded world.

- A **Program** is a sequence of steps that perform a specific functionality.

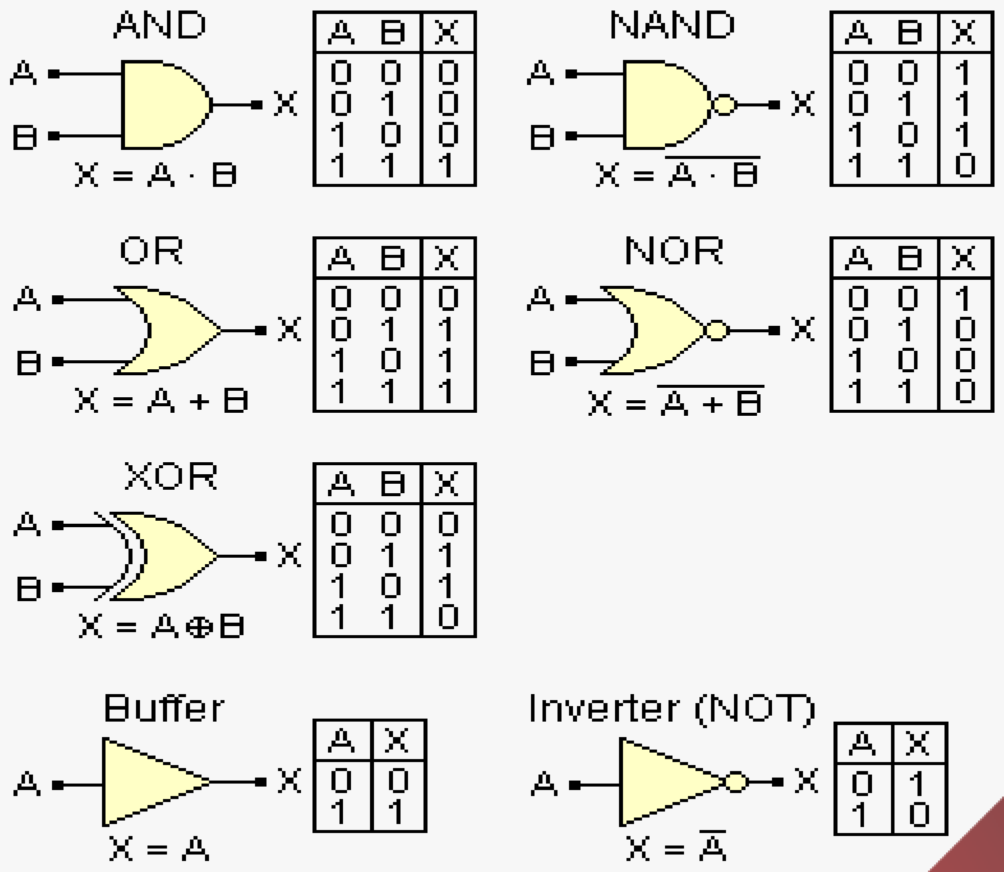


Agenda

1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

Digital design review

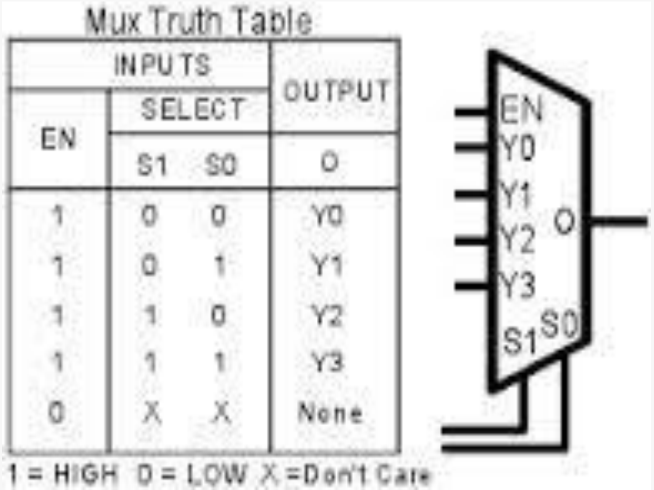
- **Logic Gates:**



Digital design review

- **Combinational logic components**

Multiplexer

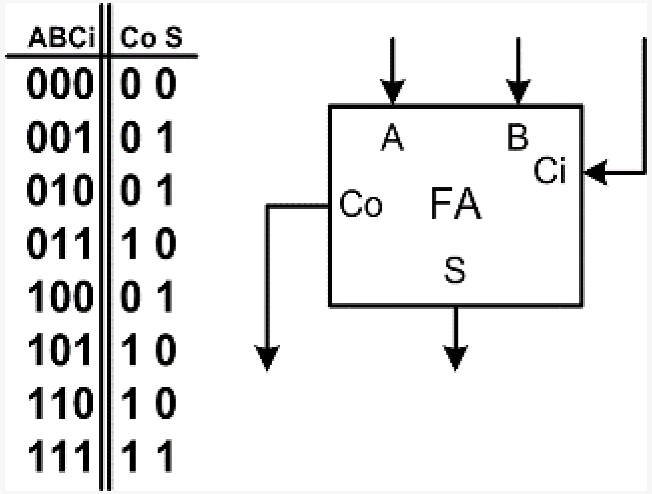


Mux Truth Table

INPUTS			OUTPUT
EN	SELECT S1	SELECT S0	O
1	0	0	Y0
1	0	1	Y1
1	1	0	Y2
1	1	1	Y3
0	X	X	None

1 = HIGH 0 = LOW X =Don't Care

Full Adder



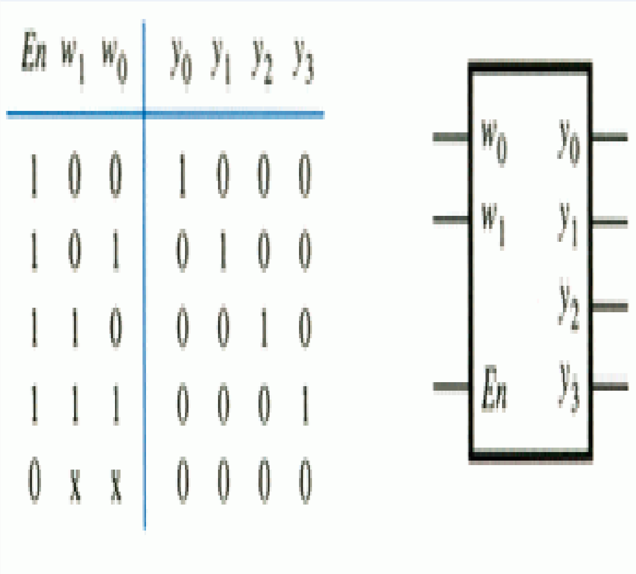
ABCi	Co S
000	0 0
001	0 1
010	0 1
011	1 0
100	0 1
101	1 0
110	1 0
111	1 1

AMIT
Beyond Education

Digital design review

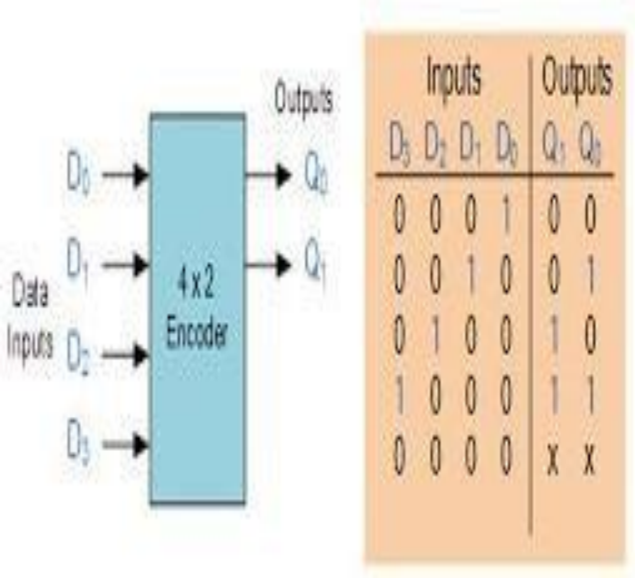
- **Combinational logic components**

Binary decoder



En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

Priority encoder

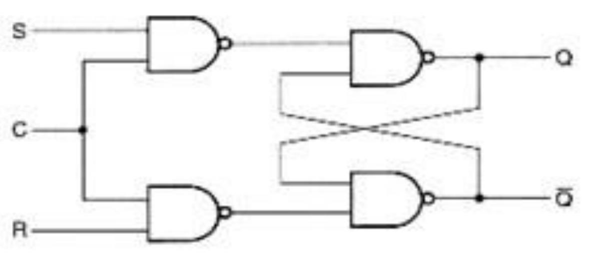


Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

Digital design review

- **Sequential logic components**

Latches: SR

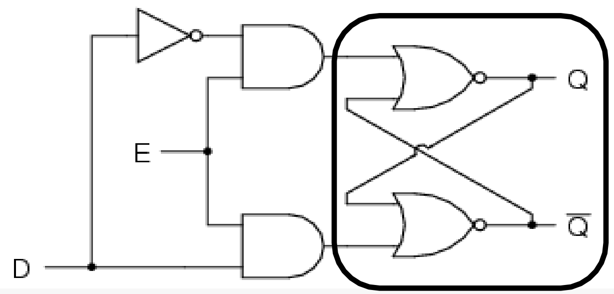


(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

Latches: D

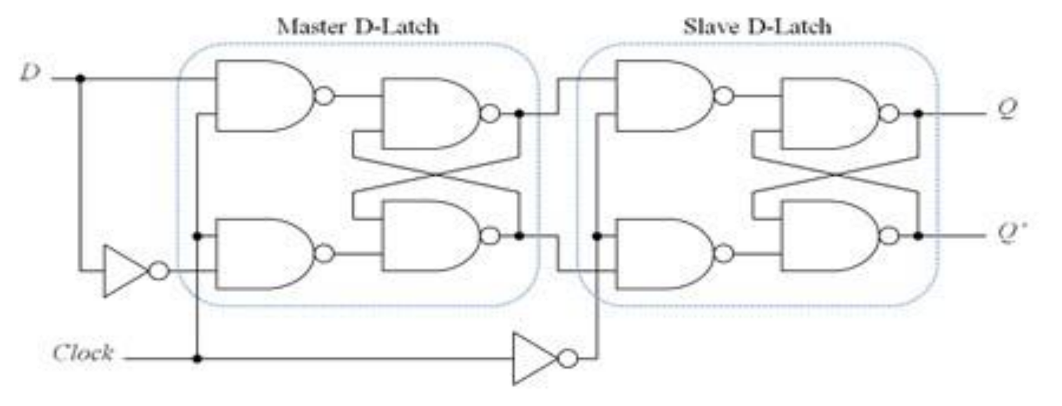


E	D	Q	$\overline{Q}$
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

Digital design review

- **Sequential logic components**

Flip-flops: D



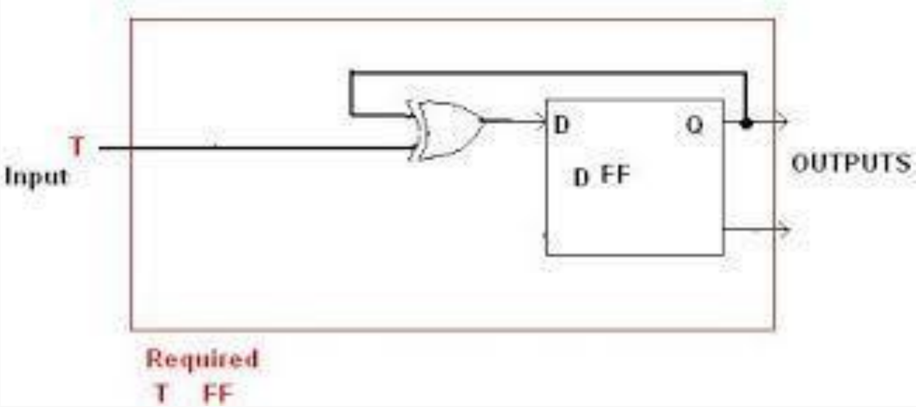
$\overline{SET}$	$\overline{RESET}$	D	CK	Q	$\overline{Q}$
0	1	-	-	1	0
1	0	-	-	0	1
0	0	-	-	1	1
1	1	1	↑	1	0
1	1	0	↑	0	1

AMIT
Beyond Education

Digital design review

- **Sequential logic components**

Flip-flops: T

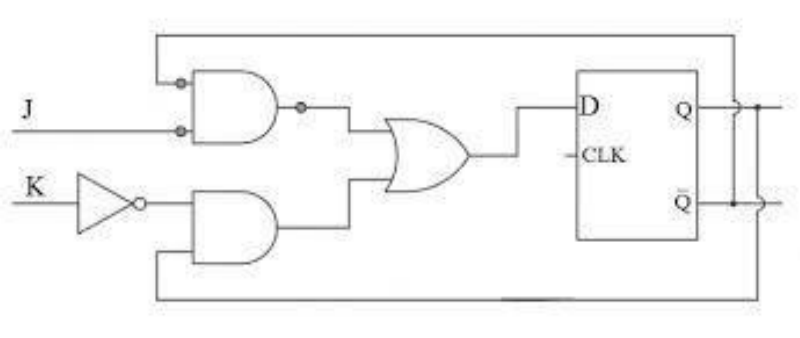


		Previous		New	
T	E	Q	Q (bar)	Q	Q (bar)
X	0	X	X	PREVIOUS VALUES	
0	1	0	1	0	1
0	1	1	0	1	0
1	1	0	1	1	0
1	1	1	0	0	1

Digital design review

- **Sequential logic components**

Flip-flops: JK

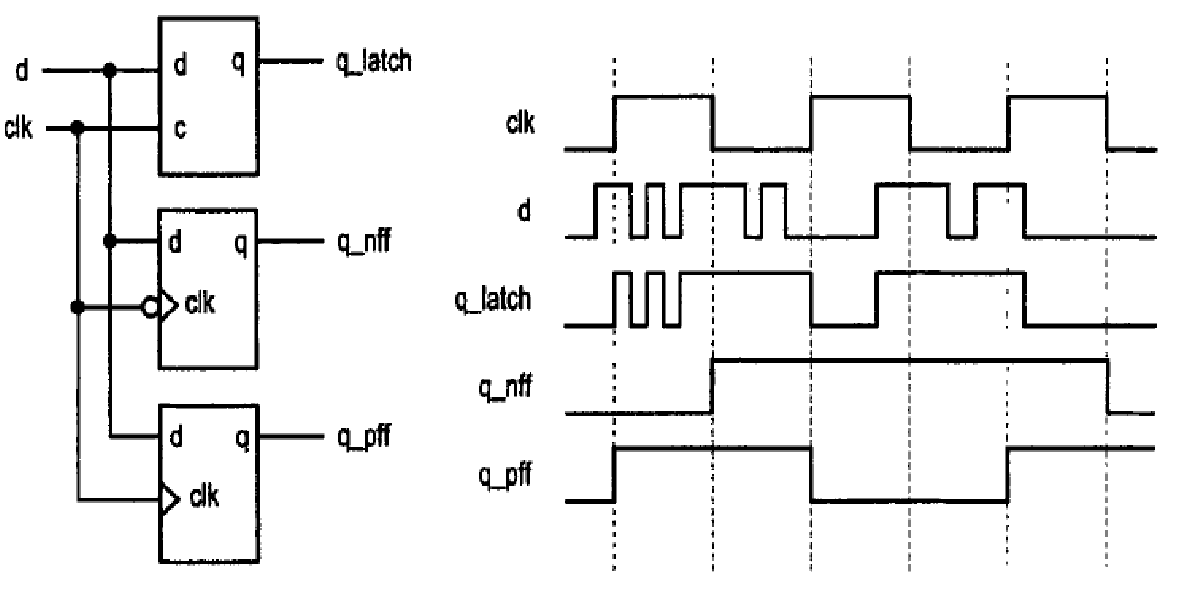


J	K	Q(t+1)
0	0	Q(t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	$\bar{Q}(t)$

AMIT Beyond Education

Digital design review

- **Level Detection Vs. Edge Detection**



Agenda

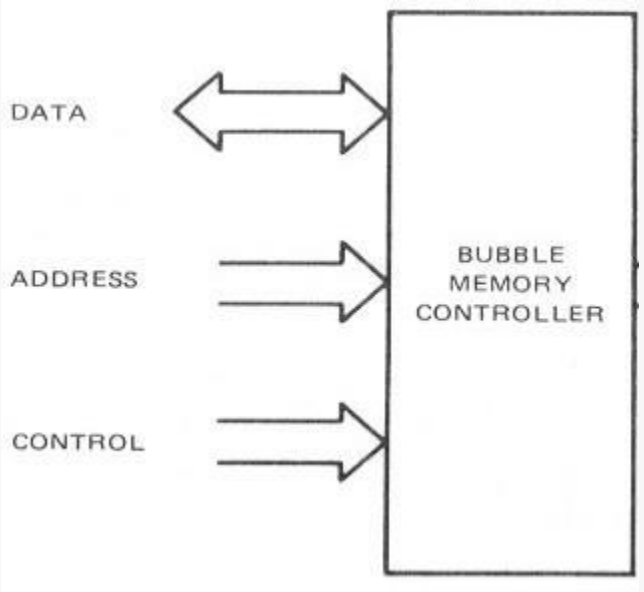
1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

AMIT
Beyond Education

Memory Types and architectures

Memory in a nutshell:

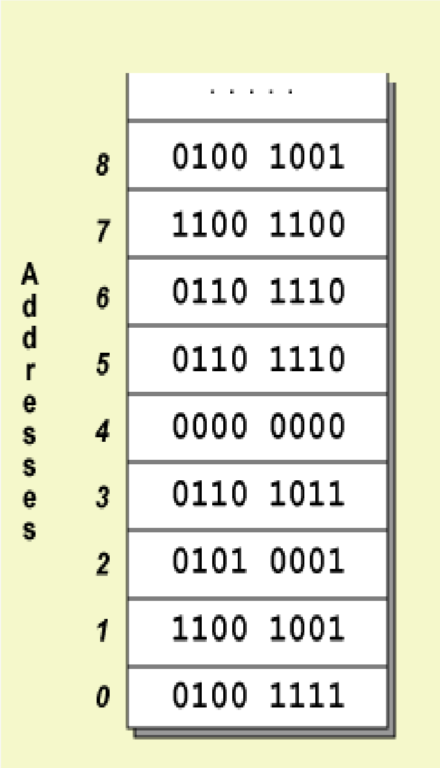
- Memory is the unit that is used to store binary data for a specific time.
- It could be read and/or written via Data Path, Address Path and Control Path.



Memory Types and architectures

•Memory in a nutshell:

- Memory is generally consisted of a group of locations, each one is selected using the applied address.
- Each one is also containing a specific data which is appeared on the Data path when it is needed.
- Each location could be bit addressable or word addressable.
- Reading and or writing is controlled using control path (CE, RE and WE).



AMIT
Beyond Education

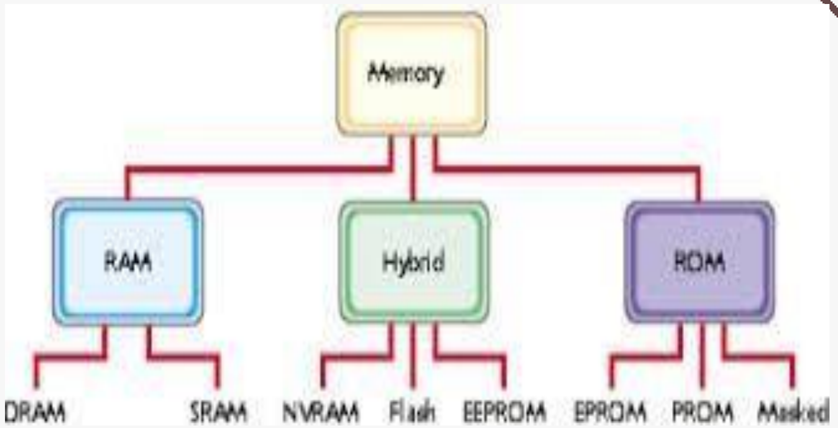
Memory Types and architectures

Memory in a nutshell:

- In terms of access method, memory could be read only memory (like ROMs and Read only CDs) or read/write memory like(RAMs, EEPROMs, Flash EEPROMs and Magnetic disks).

- In terms of data keeping with respect to supply voltage validity, Memory could be Volatile (like RAMs) and non volatile (like ROMs, FLASH EEPROMs and Magnetic Disks).

- In terms of method of access, Memory could be sequential access like magnetic taps or random access like RAMs.



Memory Types and architectures

Memory in a Details:

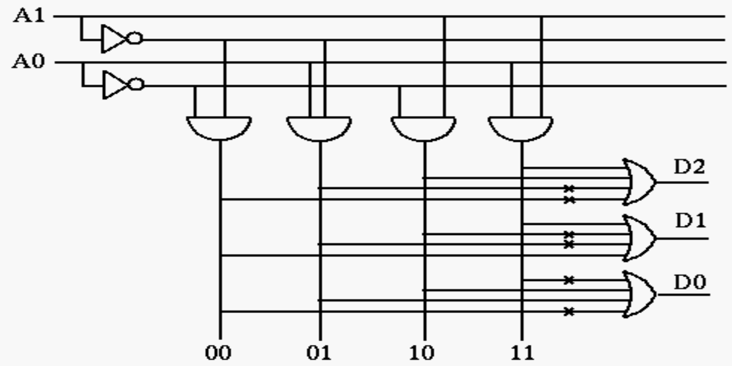
Read Only Memory (ROM):

- It is non Volatile, Random access and Read Only.
- It can be implemented as a combinational circuit because the contents of the ROM can be treated as the functions of the address.
- The information is stored in the structure and connections of the circuit, without any real storage capability actually needed.

0	010
1	001
2	101
3	110

A 4x3 ROM

Address		Data		
A_0	A_1	D_0	D_1	D_2
0	0	0	1	0
0	1	0	0	1
1	0	1	0	1
1	1	1	1	0



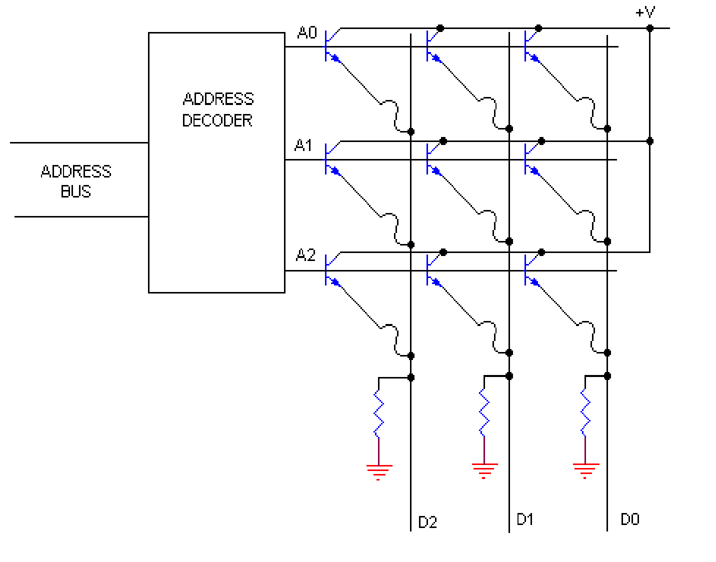
ROM implemented as a combinational circuit

Memory Types and architectures

Memory in a Details:

Programmable Read Only Memory (PROM):

- It is non Volatile, Random access and Read Only.
- It is like Rom in its architecture.
- It can only be programmed once.
- A jolt of static electricity can easily cause fuses in the PROM to burn out, changing essential bits from 1 to 0.
- it is more expensive than ROMs.

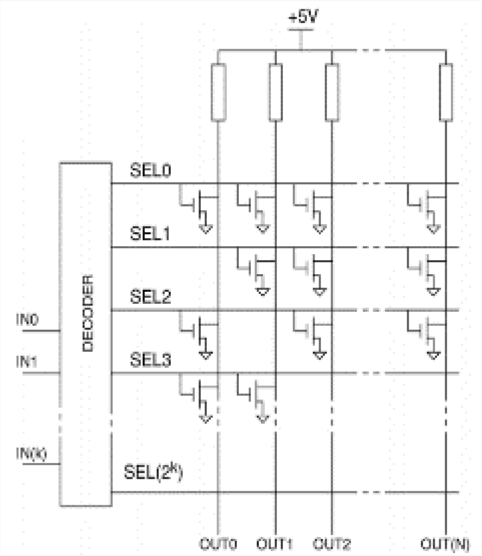


PROGRAMMABLE READ ONLY MEMORY (PROM)

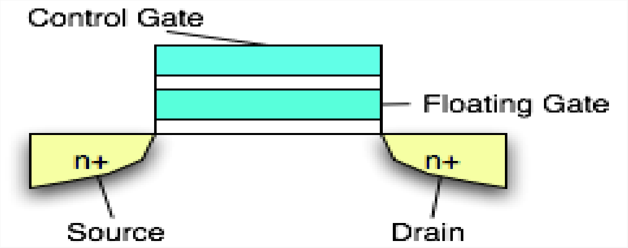
Memory Types and architectures

Memory in a Details:

Erasable Programmable Read Only Memory(EPROM):



- It is non Volatile, Random access But rewritable.
- It is an array of floating-gate transistors individually programmed by an electronic device that supplies higher voltages than those normally used in digital circuits.
- It can be erased by exposing it to strong ultraviolet light source.



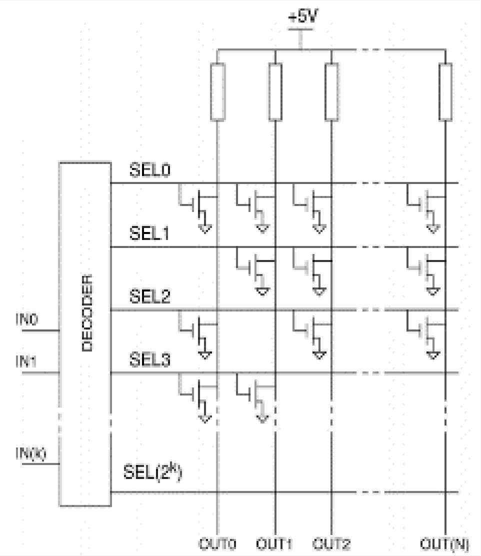
AMIT
Beyond Education

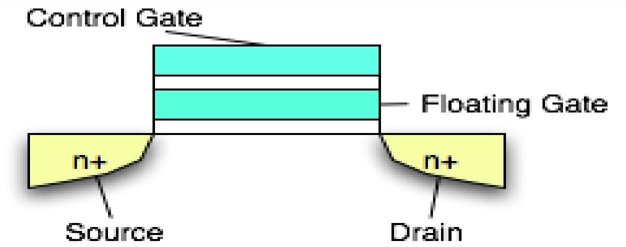
Memory Types and architectures

Memory in a Details:

Electrically Erasable Programmable Read Only Memory(EEPROM):

- It is non Volatile, Random access But rewritable.
- It is like Rom in its architecture.
- It is an array of floating-gate transistors individually programmed by an electronic device that supplies higher voltages than those normally used in digital circuits.
- It can be erased by applying electric field.





Memory Types and architectures

•Memory in a Details:

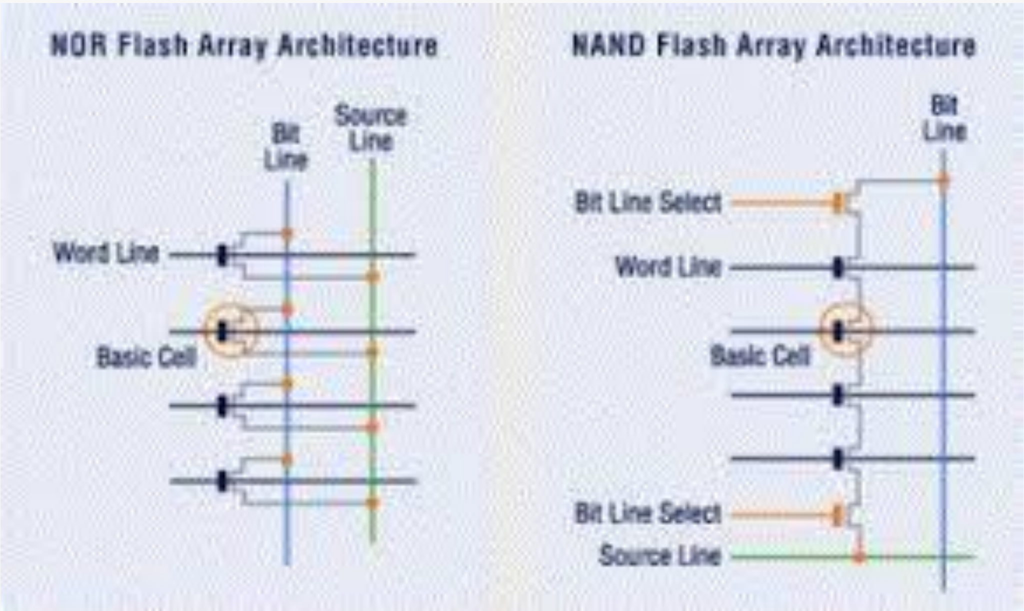
Flash Electrically Erasable Programmable Read Only Memory(Flash EEPROM):

- It is non Volatile, Random access But rewritable.
- It is made of EEPROM.
- It is not Byte Addressable like Rom, it is sector addressable.
- A sector is a group of bytes that can be accessed together.
- It has two types nand flash and nor flash.

Memory Types and architectures

Memory in a Details:

Flash Electrically Erasable Programmable Read Only Memory(Flash EEPROM):
Nand Flash VS Nor Flash



AMIT
Beyond Education

Memory Types and architectures

Memory in a Details:

Memory in a Details:
Flash Electrically Erasable Programmable Read Only Memory(Flash EEPROM):
Nand Flash VS Nor Flash

	NOR	NAND
Memory size	<= 512 Mbit	1–8 Gbit
Sector size	~1 Mbit	~1 Mbit
Output parallelism	Byte/Word/Dword	Byte/Word
Read parallelism	8–16 Word	2 Kbyte
Write parallelism	8–16 Word	2 Kbyte
Read access time	<80 ns	20 µs
Program Time	9 µs/Word	400 µs/page
Erase time	1 s/sector	1 ms/sector

AMIT
Beyond Education

Memory Types and architectures

Memory in a Details:

Random Access Memory (RAM):

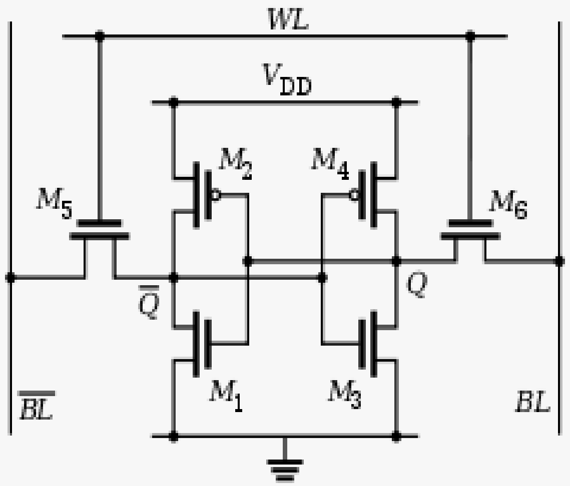
- It is Volatile, Random access and rewritable.
- Alternatively referred to as main memory, primary memory, or system memory.
- In terms of architecture, it could be Static or dynamic ram.

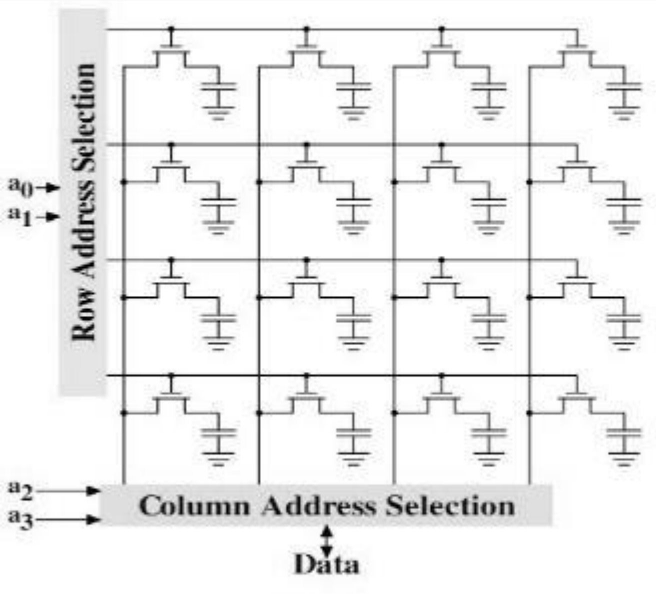
AMIT
Beyond Education

Memory Types and architectures

Memory in a Details:
Random Access Memory (RAM):

SRAM Vs DRAM





AMIT
Beyond Education

Memory Types and architectures

Memory in a Details:

Random Access Memory (RAM):
SRAM Vs DRAM

1. SRAM is static while DRAM is dynamic
2. SRAM is faster compared to DRAM
3. SRAM consumes less power than DRAM
4. SRAM uses more transistors per bit of memory compared to DRAM
5. SRAM is more expensive than DRAM
6. Cheaper DRAM is used in main memory while SRAM is commonly used in cache memory

Memory Types and architectures

Memory in a Details:
Memory Register

A register is a storage device capable of holding binary data.

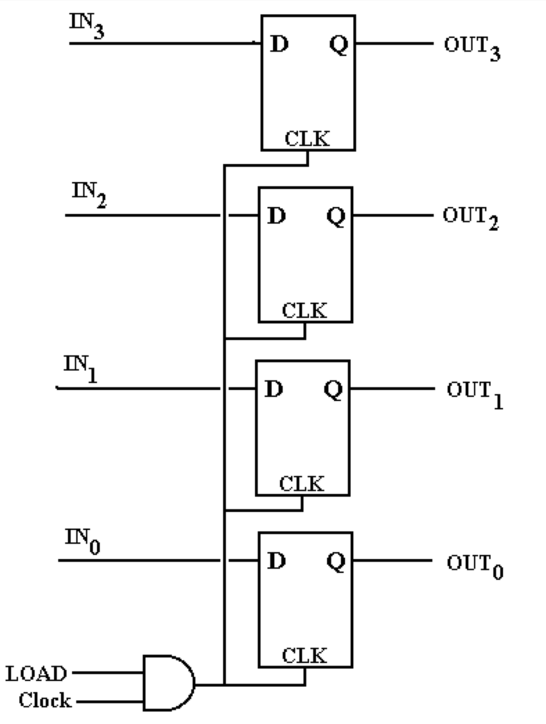
It is best viewed as a collection of flip-flops, usually D flip-flops.

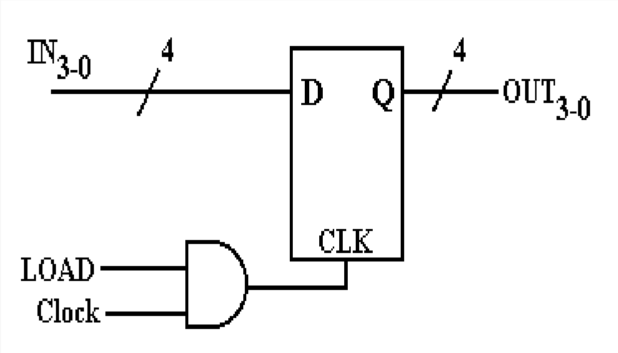
To store N bits, a register must have N flip-flops, one for each bit to be stored.

We show a design for a four-bit register with a synchronous LOAD.

Memory Types and architectures

Memory in a Details:
Memory Register





Agenda

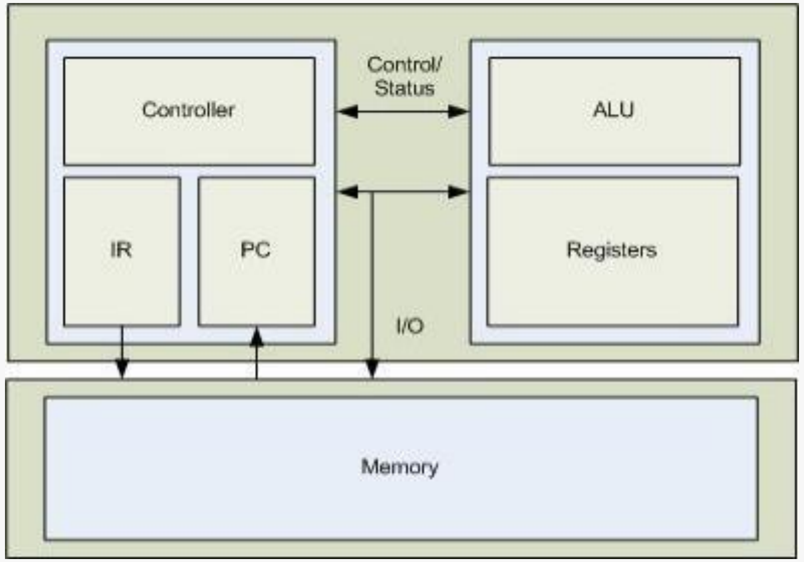
1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

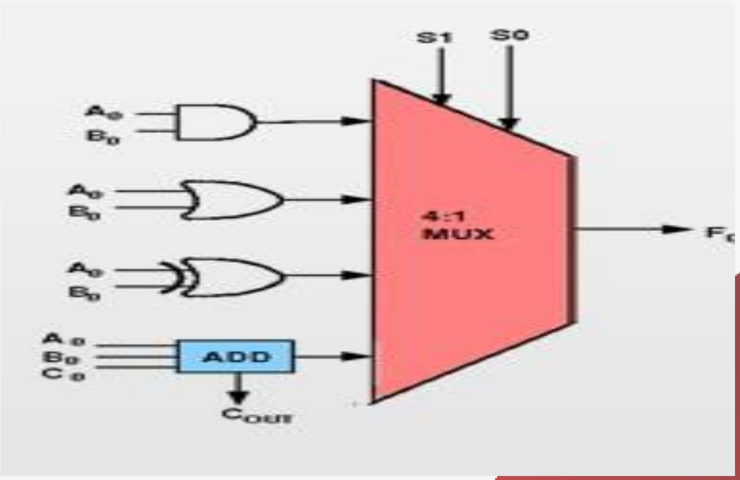
AMIT Beyond Education

Central Processing Unit architecture

•Introduction

- CPU is Consisted of three main parts ALU,CU and Registers,
- ALU: Arithmetic Logic Unit is containing the logical circuits that perform all the arithmetic and logic operations required by CPU.
- Registers: is containing the input and output argument for any operation done by ALU.
- CU: is the Control Unit which select the required operation, select the data source and destination registers.





Central Processing Unit architecture

Categories of CPU Registers:

1. **User-accessible registers:**

- instructions that can be read or written by machine instructions.
- The most common division of user-accessible registers is into data registers and address registers.

1. **Internal registers:**

- registers not accessible by instructions, used internally for processor operations.

Central Processing Unit architecture

•**Categories of CPU Registers:**

1. **User-accessible registers:**

- **Accumulator:**

  - a register in which intermediate arithmetic and logic results are stored.

  - Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) to main memory.

  -**Address registers:**

  -Registers hold addresses and are used by instructions that indirectly access primary memory.

  -for example, the **Index register** used to hold the address offset in some addressing modes and the **Stack Pointer** used to hold the Top of stack address.

Central Processing Unit architecture

Categories of CPU Registers:

1. **User-accessible registers:**

-**General Purpose Registers(GPRs):**they are combined Data/Address registers and rarely the register file is unified to include floating point as well.

-Special Purpose Registers(SPRs):

-They are registers with special functionalities.

-For example, the **Status Registers** which hold the program status (flags) and the **Program Counter** which hold the address of the instruction to be executed.

Central Processing Unit architecture

Categories of CPU Registers:

2- Internal registers:

-**Instruction Register(IR):** holding the instruction currently being executed.

-**Memory Buffer Register(MBR):**
-stores the data being transferred to and from the immediate access store.
-A data item will be copied to the MBR ready for use at the next clock cycle, when it can be either used by the processor for reading or writing or stored in main memory after being written.

Central Processing Unit architecture

-**Memory Data Register(MDR):**

-It acts like a buffer and holds anything that is copied from the memory ready for the processor to use it.

Central Processing Unit architecture

Memory Address Registers(MAR):

-It holds the memory location of data that needs to be accessed.

-When reading from memory, data addressed by MAR is fed into the MDR and then used by the CPU.

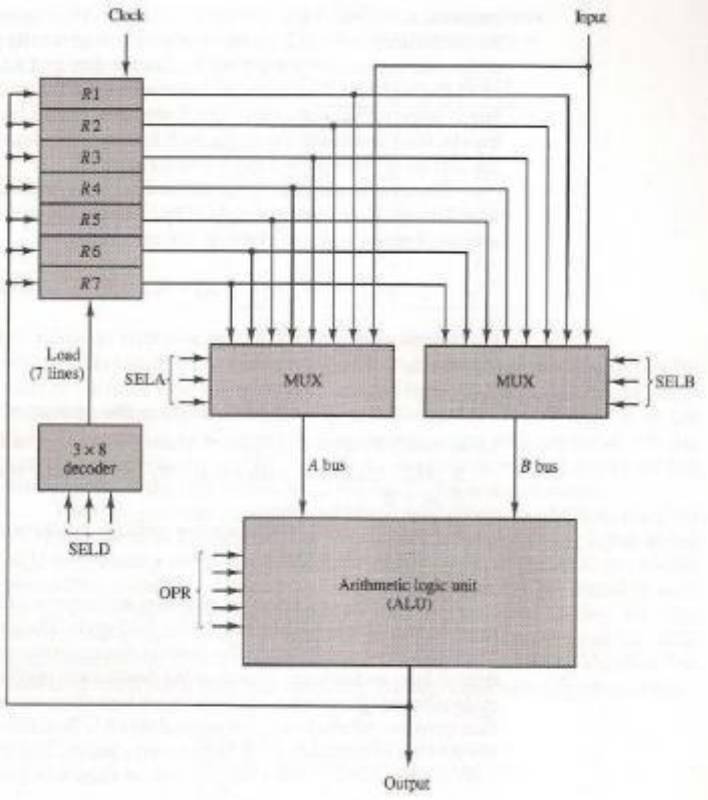
- When writing to memory, the CPU writes data from MDR to the memory location whose address is stored in MAR.

Central Processing Unit architecture

General CPU Organization:

1- BUS System:

- This is an architecture of seven registers.
- A decoder is used to select which register to be written.
- Two multiplexers are used to select which registers should be read to be the two ALU operands.
- Number of address bits to select is log2(number of registers)=3.



Central Processing Unit architecture

General CPU Organization:

2- Control Word:

- The Control Word is generally the instruction that will be executed by the CPU hardware.
- It is consisted of four fields.
- SELA and SELB for source registers selection, they are applied by the control unit to the two multiplexers.
- SELD to select the destination register, it is applied by the control unit to the decoder.
- OPR is selection which operation will be done by ALU.

TABLE 8-1 Encoding of Register Selection Fields

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	$R1$	$R1$	$R1$
010	$R2$	$R2$	$R2$
011	$R3$	$R3$	$R3$
100	$R4$	$R4$	$R4$
101	$R5$	$R5$	$R5$
110	$R6$	$R6$	$R6$
111	$R7$	$R7$	$R7$

AMIT Beyond Education

Central Processing Unit architecture

General CPU Organization:

3-ALU

- ALU is the unit that performs the arithmetic, logical and in some cases shift operations required by the software program.
- It is consisted of a group of arithmetic and logical high speed circuits.
- These circuits are processing the ALU operands and generate different operations results.
- These results are selected to be generated from ALU by using the OPR field of the control word.

TABLE 8-2 Encoding of ALU Operations

OPR Select	Operation	Symbol
00000	Transfer *A*	TSFA
00001	Increment *A*	INCA
00010	Add *A* + *B*	ADD
00101	Subtract *A* − *B*	SUB
00110	Decrement *A*	DECA
01000	AND *A* and *B*	AND
01010	OR *A* and *B*	OR
01100	XOR *A* and *B*	XOR
01110	Complement *A*	COMA
10000	Shift right *A*	SHRA
11000	Shift left *A*	SHLA

Central Processing Unit architecture

Micro operations examples:

$$R1 \leftarrow R2 - R3$$

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control word:	010	011	001	00101

TABLE 8-3 Examples of Microoperations for the CPU

Microoperation	SELA	SELB	SELD	OPR	Control Word
$R1 \leftarrow R2 - R3$	R2	R3	R1	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	R4	R5	R4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	R6	—	R6	INCA	110 000 110 00001
$R7 \leftarrow R1$	R1	—	R7	TSFA	001 000 111 00000
Output $\leftarrow R2$	R2	—	None	TSFA	010 000 000 00000
Output $\leftarrow$ Input	Input	—	None	TSFA	000 000 000 00000
$R4 \leftarrow$ shl $R4$	R4	—	R4	SHLA	100 000 100 11000
$R5 \leftarrow 0$	R5	R5	R5	XOR	101 101 101 01100

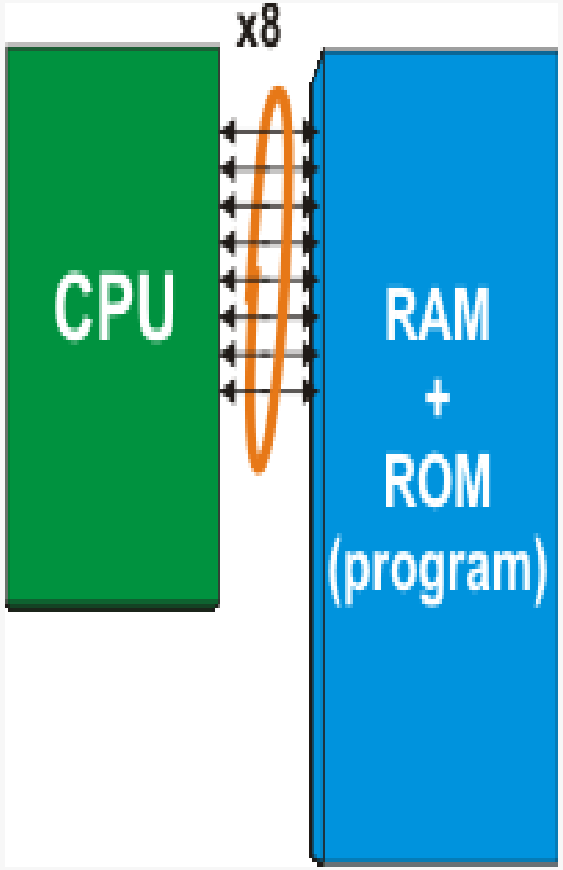
Beyond Education

Central Processing Unit architecture

General CPU Architectures

VonNeuman Architecture.

- *Processor* that uses this architecture has only one block of memory and one data bus (8-bit).
- Because of all data exchange using this 8 traffics, the bus will be overload, the communication will be very slow and not efficient.
- CPU can read instruction or read/write data from/to memory.

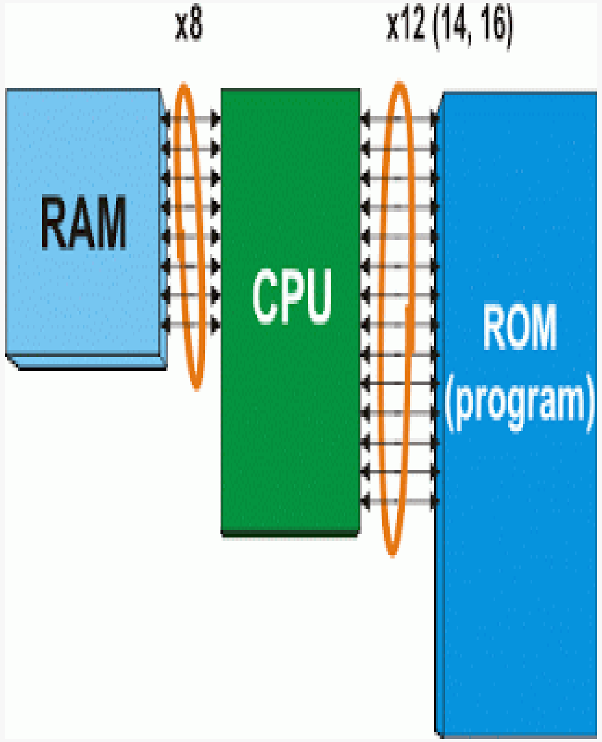


Central Processing Unit architecture

General CPU Architectures

Harvard Architecture.

- *processor* that uses this architecture has two different busses. One bus (8-bits) and connecting CPU to RAM.
- the other bus consist of some traffics (12, 14, and 16) and connecting CPU to ROM.
- CPU can read instruction and access data memory at same time.



AMIT
Beyond Education

Central Processing Unit architecture

General CPU Architectures

CISC Vs. RISC Architecture.

CISC	RISC
Emphasis on hardware	Emphasis on software
Including multi-clock complex instructions	Single-clock, just small amount instructions
Memory-to-memory : "LOAD" and "STORE" cooperate each other	Register-to-register : "LOAD" and "STORE" are separate instructions
Code size is small, slow speed	Code size is big, high speed
Transistor is used to store complex instructions	Transistor common used for register memory

Central Processing Unit architecture

General CPU Architectures

CISC Vs. RISC Architecture.

Examples of CISC

- Processor system/360
- Processor VAX
- Processor PDP-11
- CPU AMD
- Intel x86

Example of RISC

- Computer Vector
- Microprocessor Intel 960
- Itanium (IA64) from Intel Corporation
- Power PC from International Business Machine
- AVR
- ARM

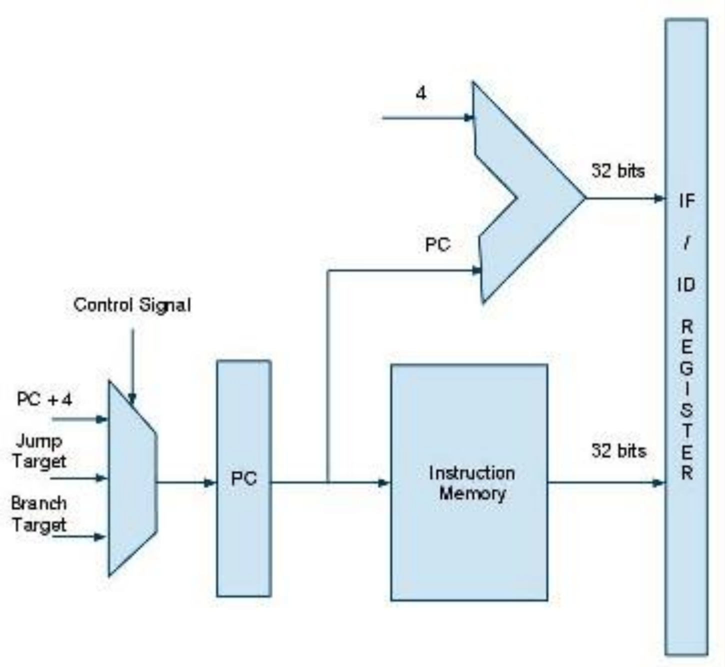
AMIT
Beyond Education

Central Processing Unit architecture

CPU Instruction Cycle

1. **Instruction Fetch:**

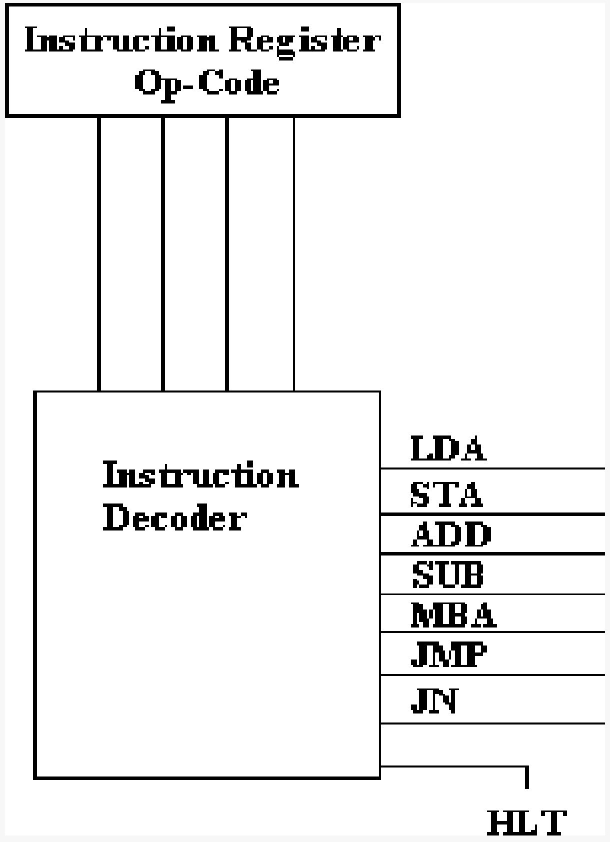
The next instruction is fetched from the memory address that is currently stored in the program counter (PC), and stored in the Instruction Register (IR). At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.



Central Processing Unit architecture

- **CPU Instruction Cycle**

2. Instruction Decode: The decoder interprets the instruction. During this cycle the instruction inside the IR (instruction register) gets decoded.



AMIT
Beyond Education

Central Processing Unit architecture

CPU Instruction Cycle

3. Instruction Execute:

The execute unit will perform the following:

- Place to store instructions and data Memory
- Perform arithmetic and logical operations Processing unit
- Determine next instruction to execute Control unit
- Get data into computer to manipulate Input devices
- Display results to user Output devices

Central Processing Unit architecture

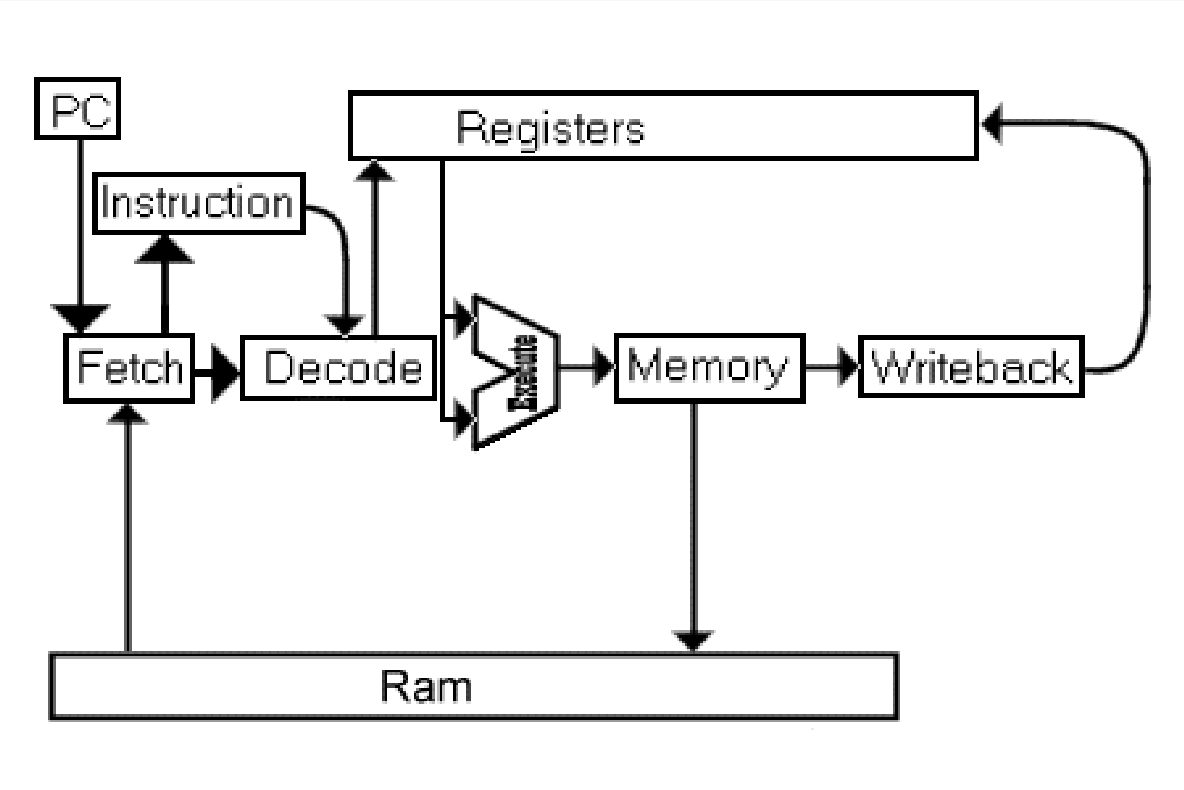
CPU Instruction Cycle

3. Instruction memory operation:
If data memory needs to be accessed, it is done so in this stage

.4. Instruction write back operation:
During this stage, both single cycle and two cycle instructions write their results into the register file.

Central Processing Unit architecture

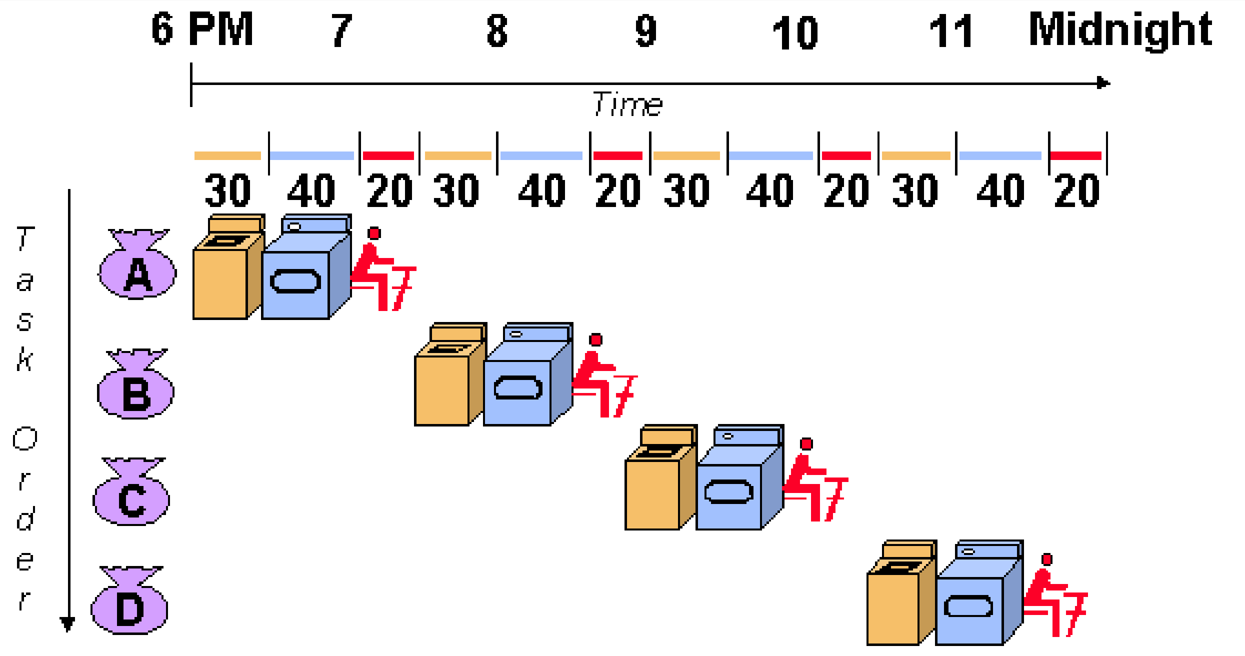
CPU Instruction Cycle



Central Processing Unit architecture

CPU Pipelined architecture:

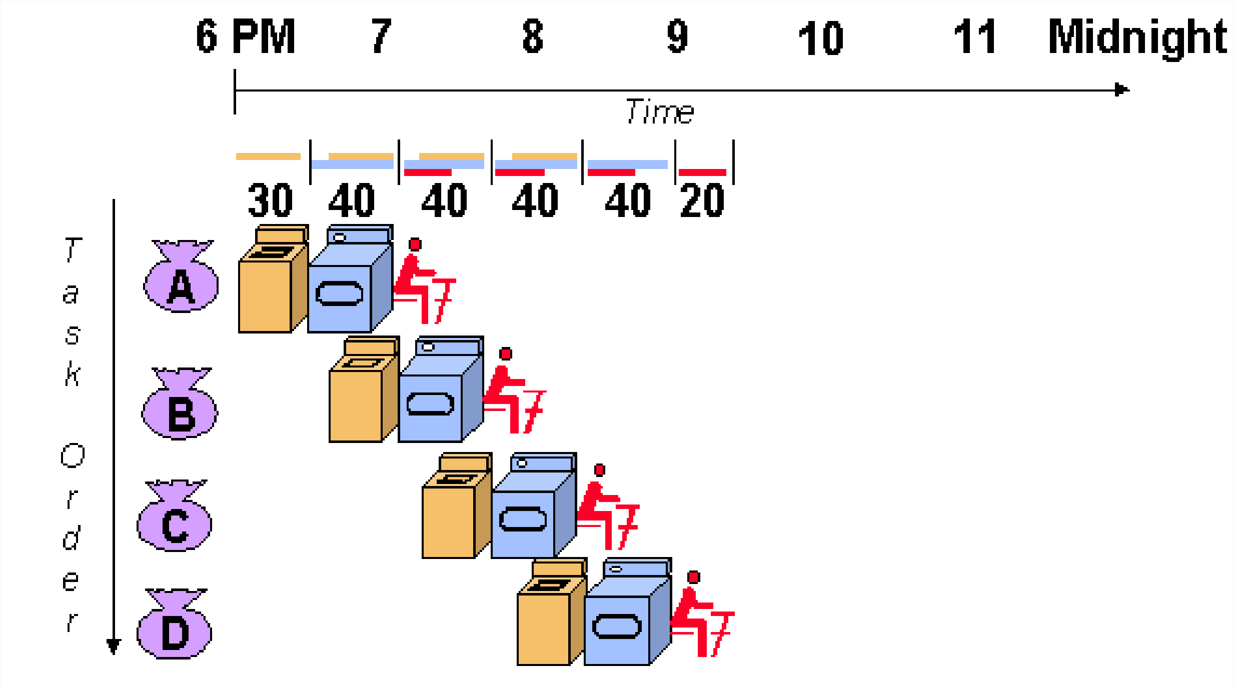
How Pipelining Works?



Central Processing Unit architecture

CPU Pipelined architecture:

How Pipelining Works?

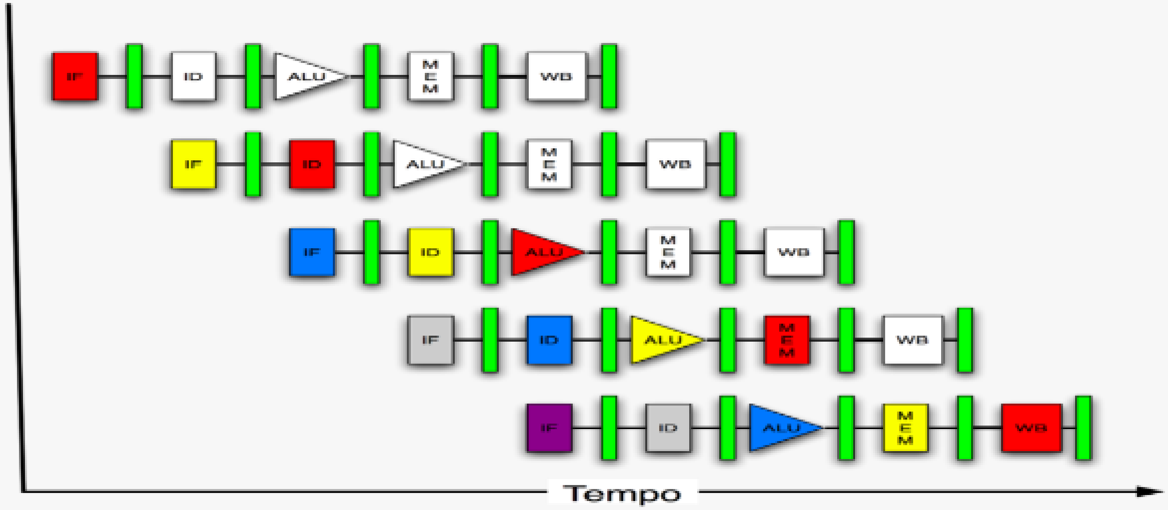


Central Processing Unit architecture

CPU Pipelined architecture:

Pipelining In microprocessor:

- Pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time.



Central Processing Unit architecture

- **CPU Pipelined architecture:**
- **Pipelining Hazard:**

Hazard prevent next instruction from executing during its designated clock cycle

– **Structural Hazard:**

HW cannot support this combination of instructions

– **Data Hazard:**

Instruction depends on result of prior instruction stil in the pipeline

– **Control Hazard:**

Pipelining of branches & other instructions that change the PC

- Common solution is to stall the pipeline until the hazard is resolved, inserting one or more "bubbles" in the pipeline

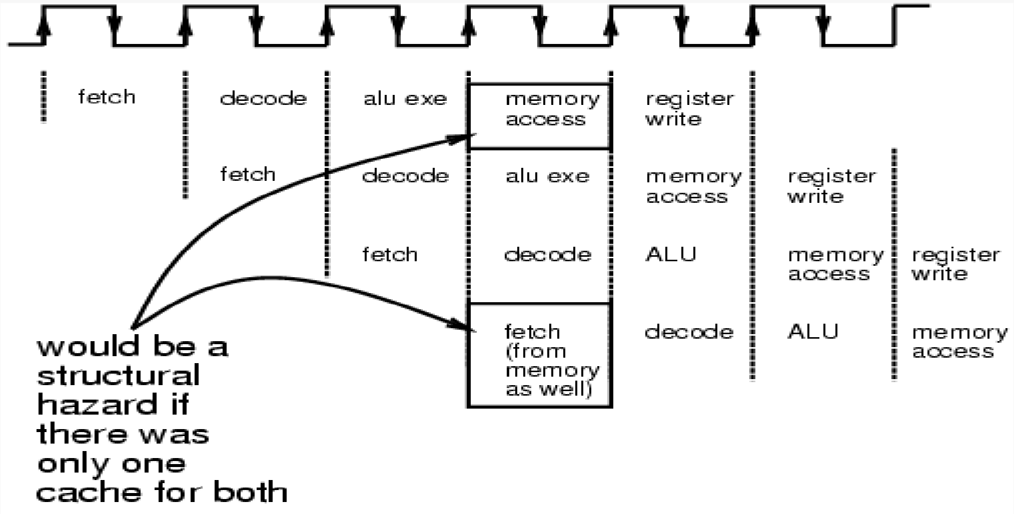
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Structural Hazard:

•A structural hazard would for example result from memory access of instruction fetch and memory access of data, were it not for separate data and instruction caches



AMIT Beyond Education

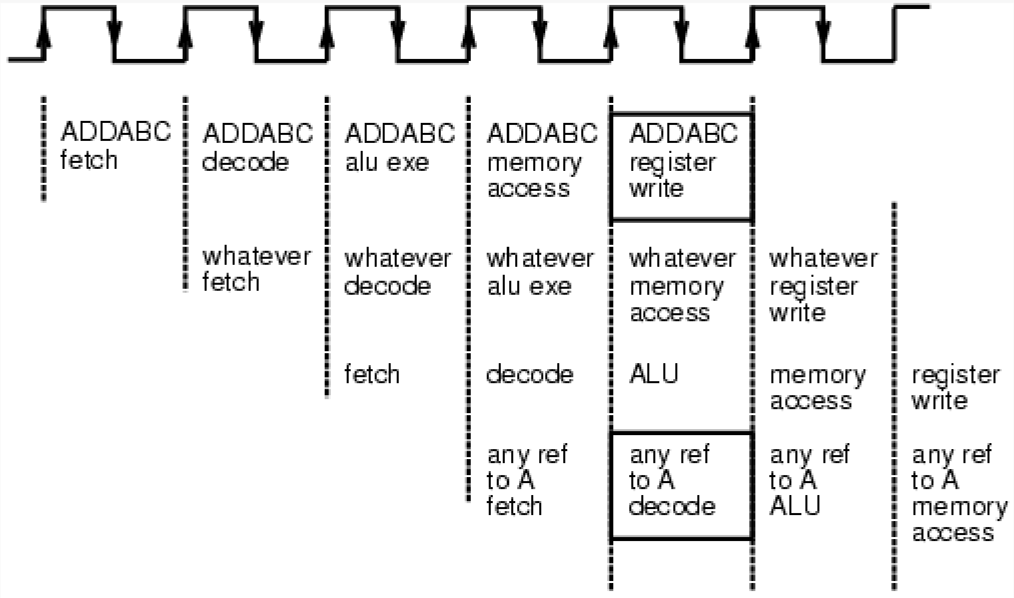
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Structural Hazard:

another example of a structural hazard is when decoding (setting up input registers) makes reference to same register as a register write:

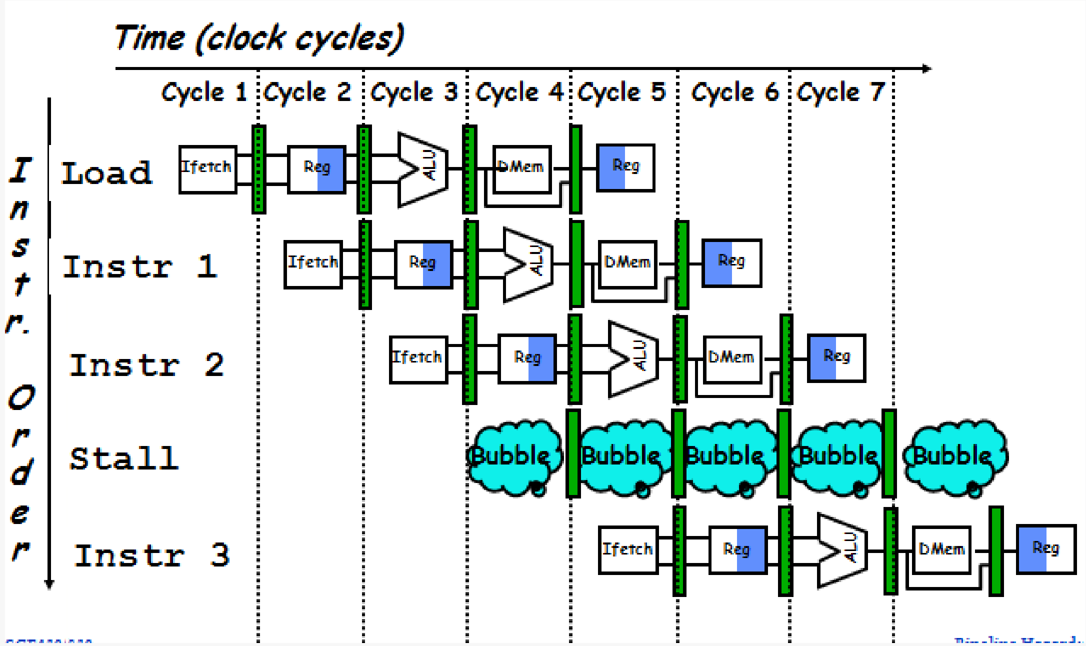


Central Processing Unit architecture

CPU Pipelined architecture:

•**Pipelining Hazard:**

– Structural Hazard(problem solutions) Stall:

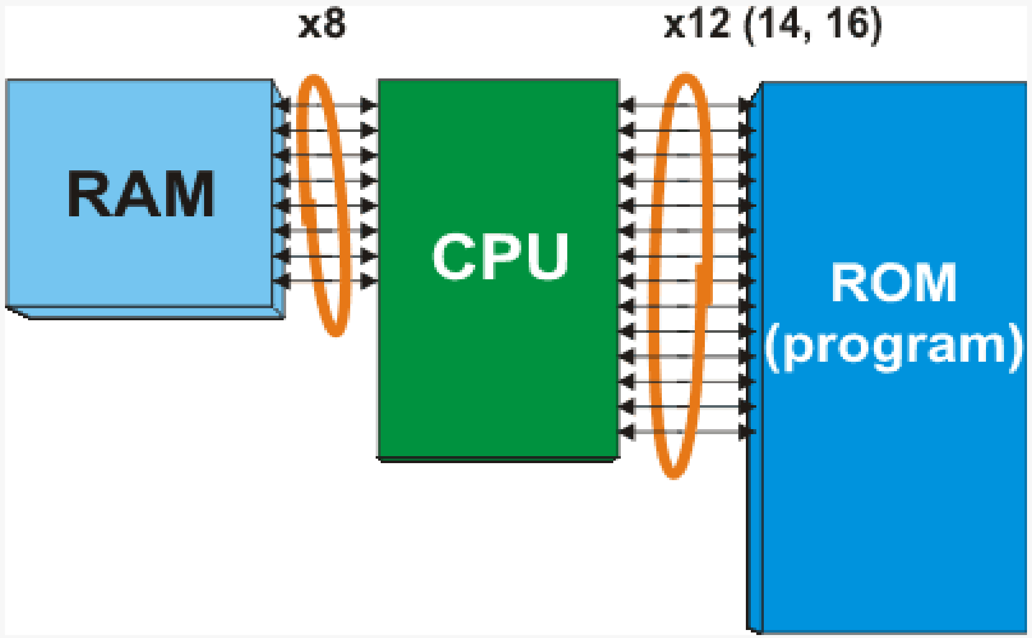


Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Structural Hazard(problem solutions) Replicate resource :



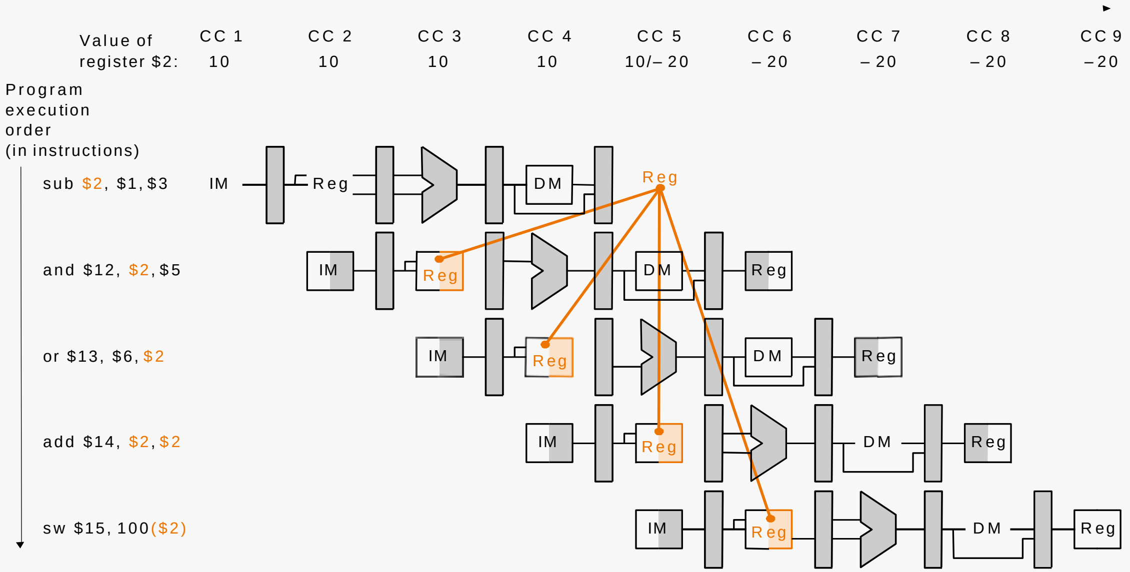
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Data Hazard:

The use of the result of the SUB instruction in the next three instructions causes a data hazard, since the register $2 is not written until after those instructions read it.



Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Read After Write (RAW)

InstrJ tries to read operand before InstrI writes it

```
I: add r1,r2,r3
J: sub r4,r1,r3
```

•Caused by a "Dependence" (in compiler nomenclature). This hazard results from an actual need for communication.

Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Read After Write (RAW)

InstrJ tries to read operand before InstrI writes it

Gets wrong operand

```
I:  sub r4,r1,r3
J:  add r1,r2,r3
K:  mul r6,r1,r7
```

Called an “anti-dependence” by compiler writers

This results from reuse of the name “r1”.

Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Read After Write (RAW)

InstrJ tries to read operand before InstrI writes it

Leaves wrong result (InstrI not InstrJ)

```
I:  sub r1,r4,r3
J:  add r1,r2,r3
K:  mul r6,r1,r7
```

Called an “output dependence” by compiler writers

This also results from the reuse of name “r1”.

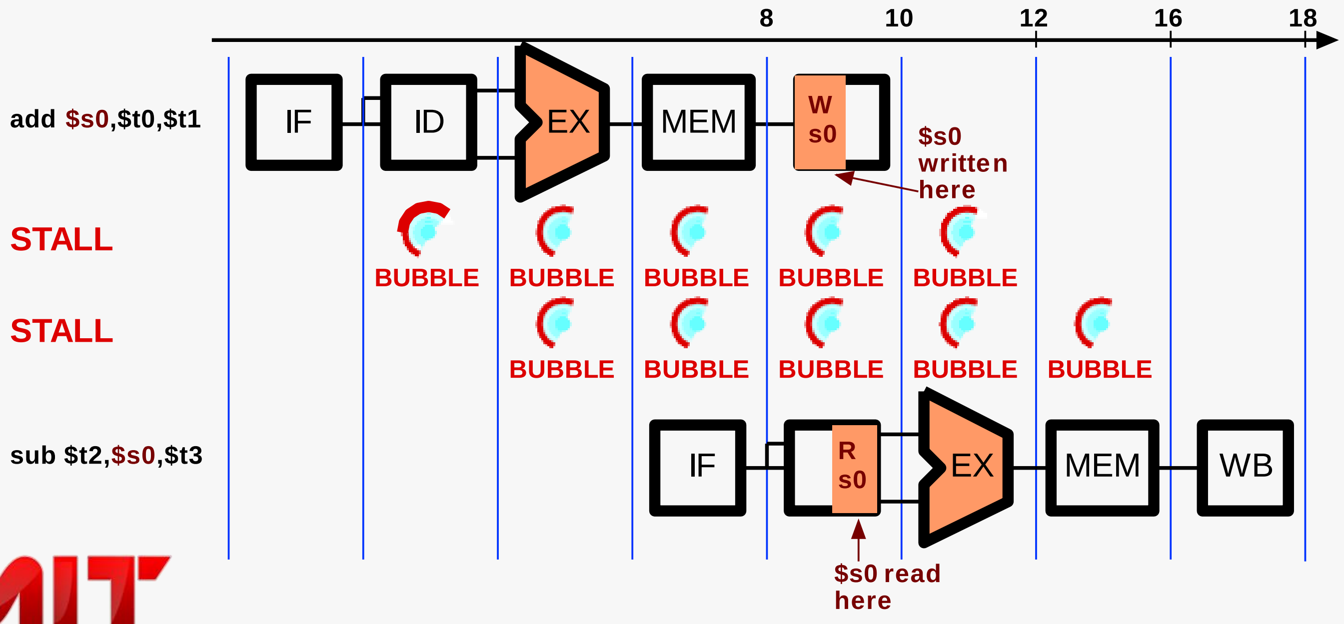
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Data hazard: Problem solution:

-Stall:



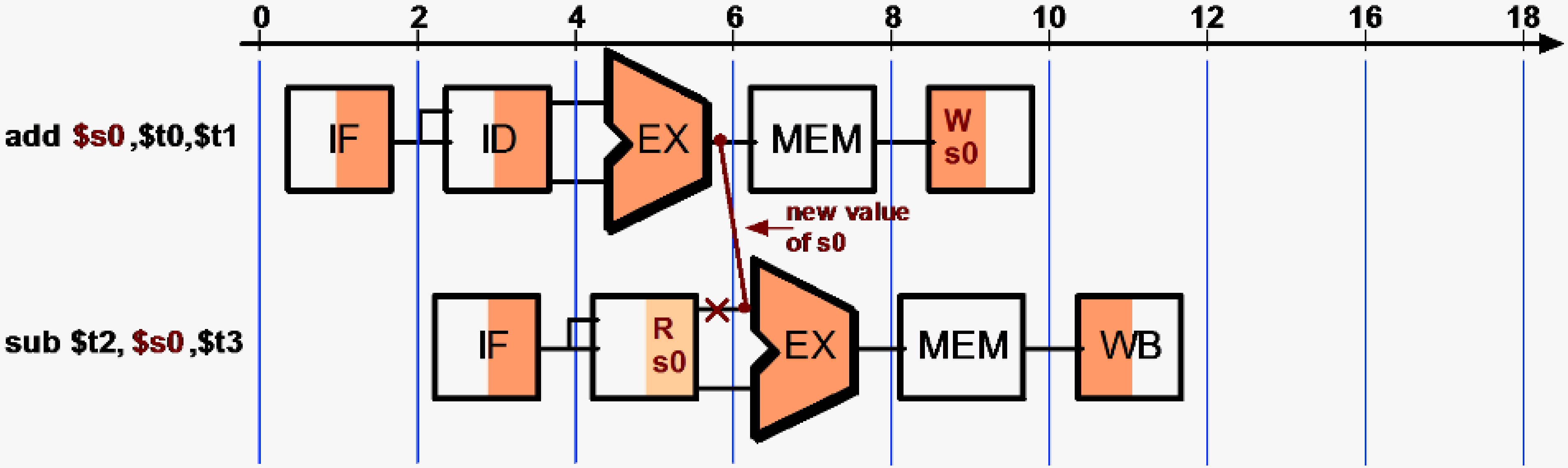
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Data hazard: Problem solution:

-Forwarding:



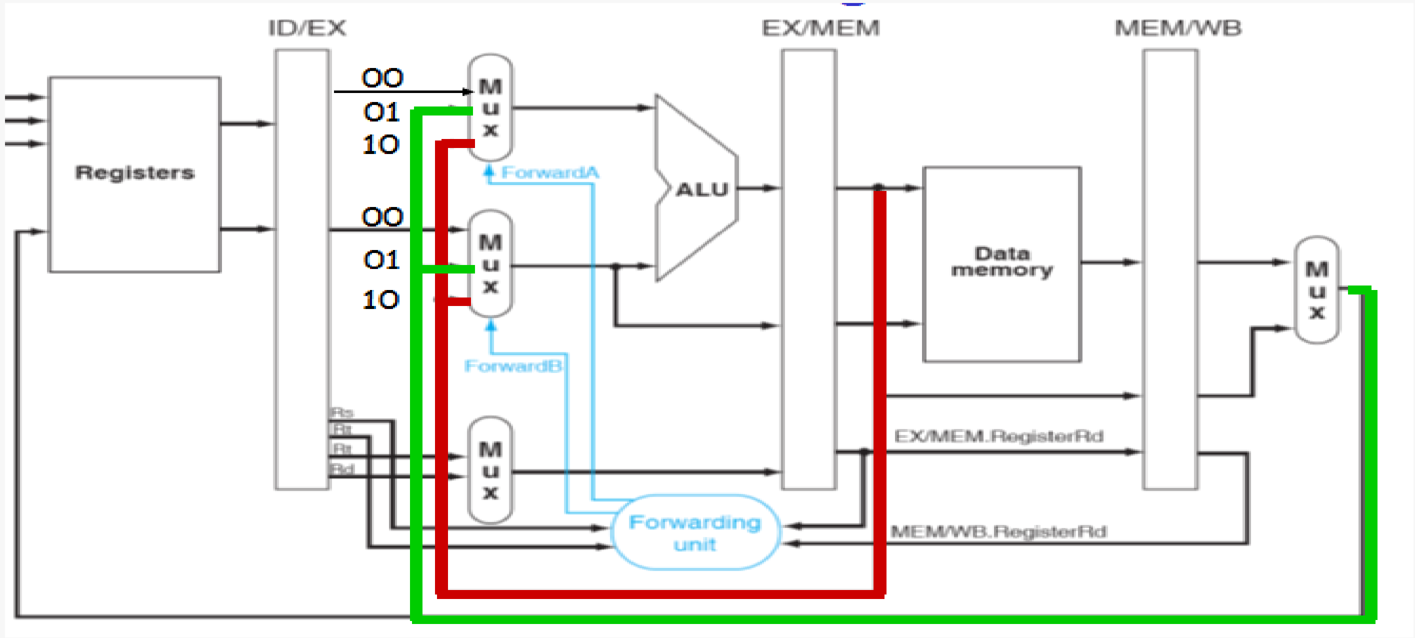
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Data hazard: Problem solution:

-Forwarding:



Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard:

A control hazard is when we need to find the destination of a branch, and can’t fetch any new instructions until we know that destination.

A branch is either

Taken: PC <= PC + 4 + Immediate

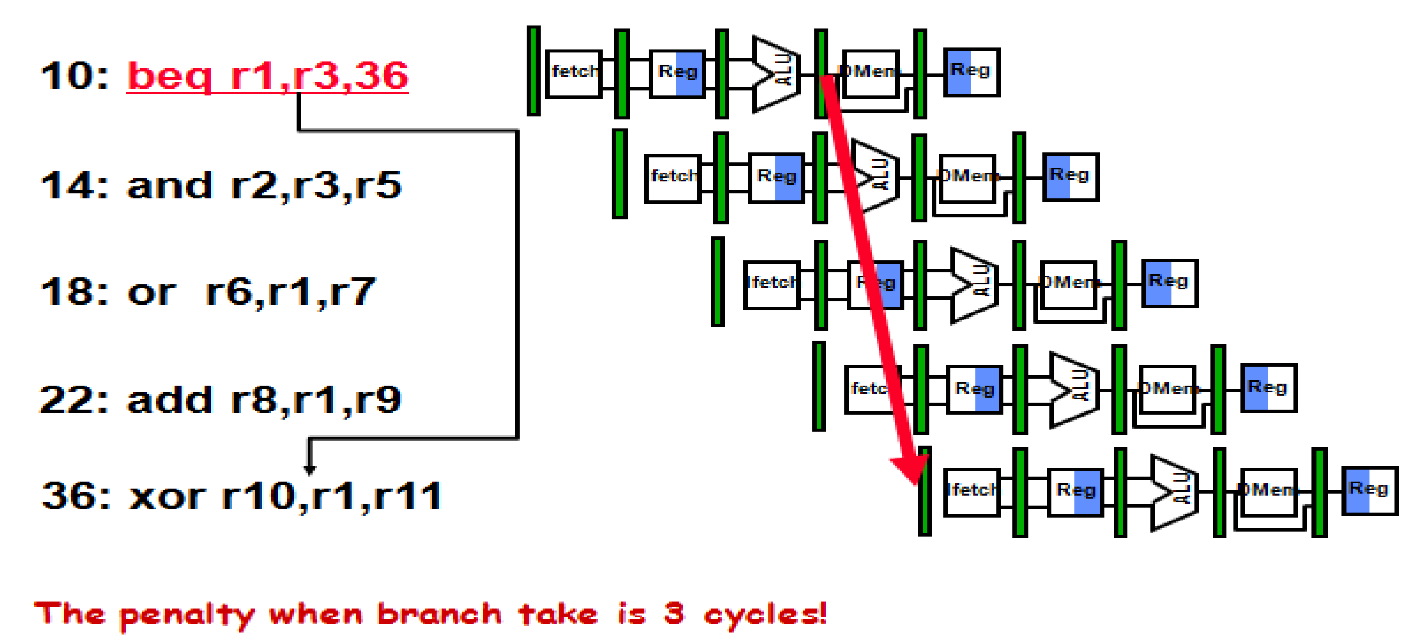
Not Taken: PC <= PC + 4

Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard:



Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard: Problem Solution

Stall

stop loading instructions until result is available

Predict

assume an outcome and continue fetching (undo if prediction is wrong)

lose cycles only on mis-prediction

Delayed branch

specify in architecture that the instruction immediately following branch is always executed

AMIT Beyond Education

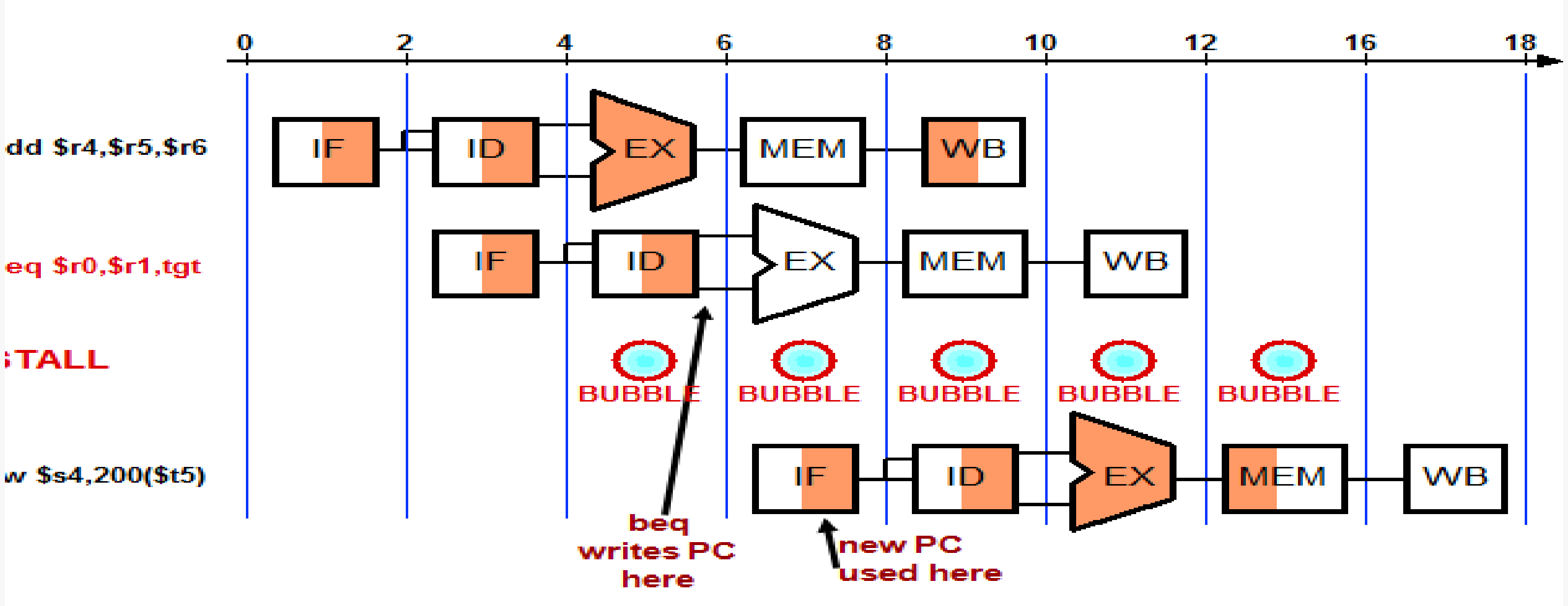
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard: Problem Solution

Stall



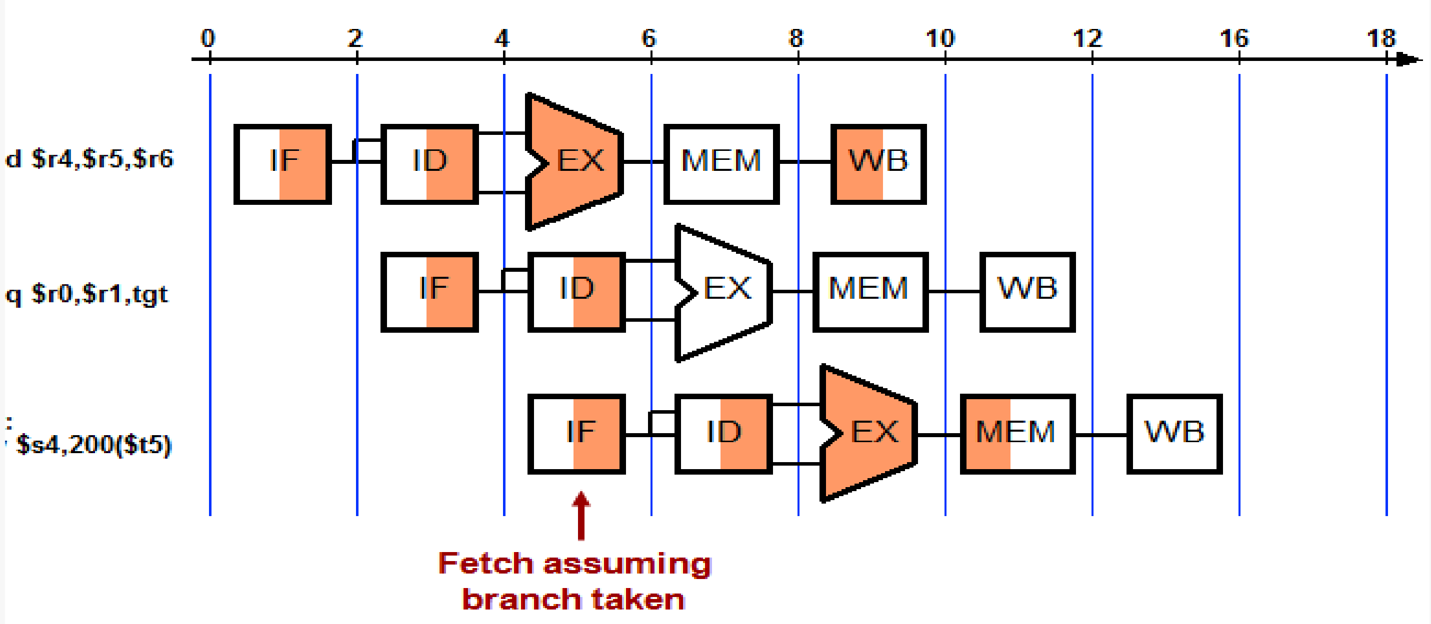
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard: Problem Solution

Prediction: Correct Prediction



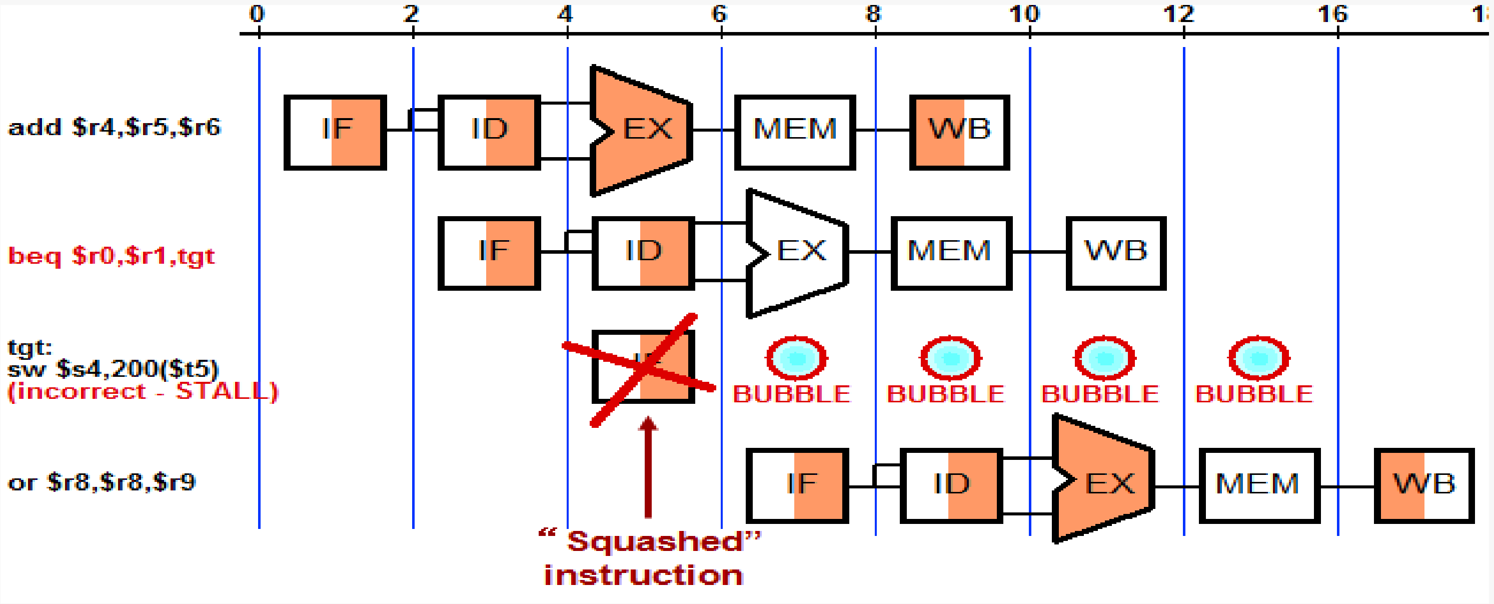
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard: Problem Solution

Prediction: Incorrect Prediction



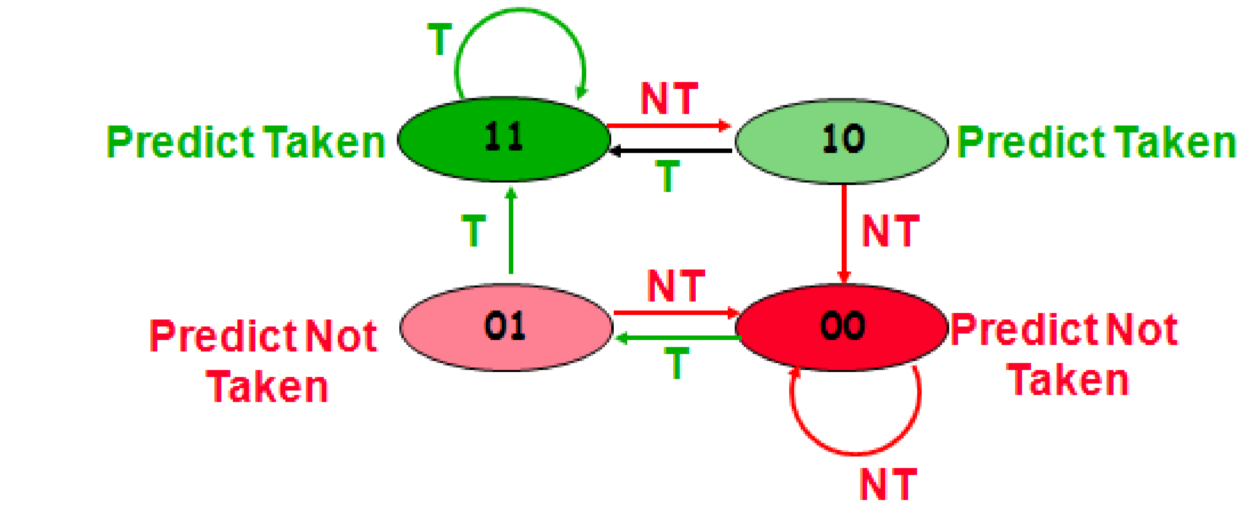
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

– Control Hazard: Problem Solution

Prediction: Prediction Example:2- bit Prediction



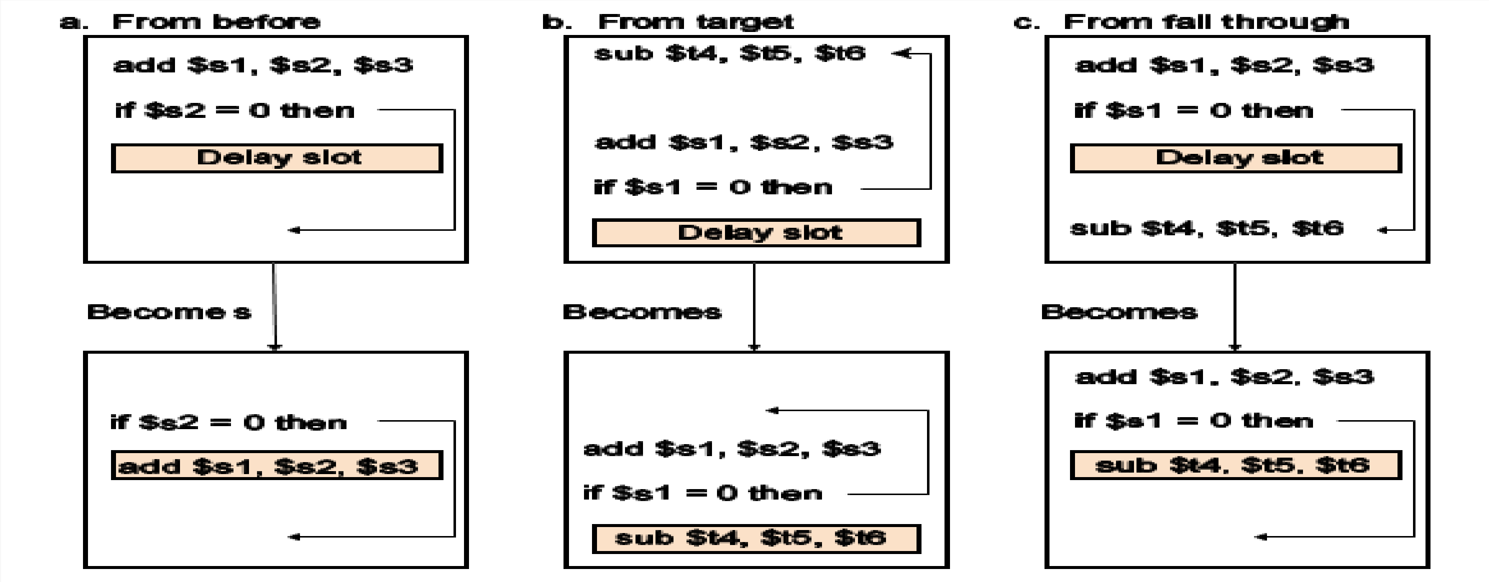
Central Processing Unit architecture

CPU Pipelined architecture:

•Pipelining Hazard:

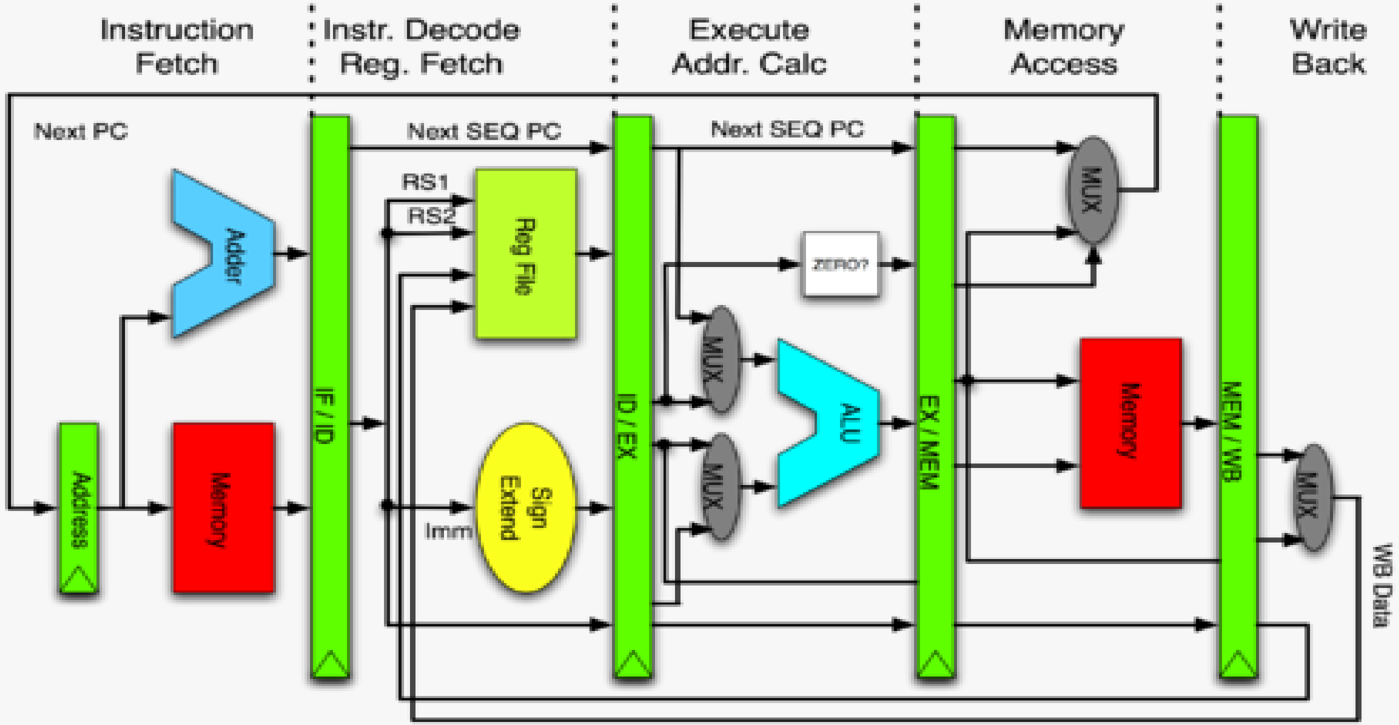
– Control Hazard: Problem Solution

Delayed Branch:



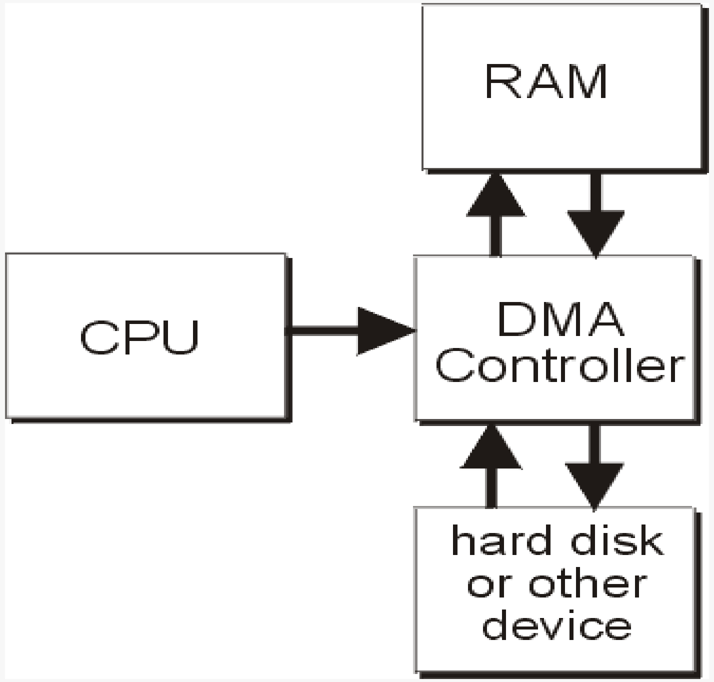
Central Processing Unit architecture

CPU Pipelined architecture:
Overall Architecture:



Central Processing Unit architecture

Direct Memory access:



Central Processing Unit architecture

•Memory addressing modes:

-The addressing mode specifies a rule of interpreting or modifying the address field of an instruction.

- The Addressing modes are used to accommodate one or both of the following provisions.

1Providing facility of pointers to memory, counters for loop control, indexing of data and program relocation.

- Reducing the number of bits in the address field of an instruction.

-A mode filed could be added to the control word to specify which addressing mode will be used.

Opcode	Mode	Address

Central Processing Unit architecture

Memory addressing modes:

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	$AC \leftarrow M[ADR]$
Indirect address	LD @ADR	$AC \leftarrow M[M[ADR]]$
Relative address	LD $ADR	$AC \leftarrow M[PC + ADR]$
Immediate operand	LD #NBR	$AC \leftarrow NBR$
Index addressing	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Register	LD R1	$AC \leftarrow R1$
Register indirect	LD (R1)	$AC \leftarrow M[R1]$
Autoincrement	LD (R1)+	$AC \leftarrow M[R1],\ R1 \leftarrow R1 + 1$

Agenda

1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

AMIT Beyond Education

Programming basic Computer

Data Transferee Instructions:

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Programming basic Computer

Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

AMIT
Beyond Education

Programming basic Computer

Logical Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

AMIT
Beyond Education

Programming basic Computer

Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

Programming basic Computer

Program Control Instructions

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

Programming basic Computer

Conditional Branch Instructions

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	$Z = 1$
BNZ	Branch if not zero	$Z = 0$
BC	Branch if carry	$C = 1$
BNC	Branch if no carry	$C = 0$
BP	Branch if plus	$S = 0$
BM	Branch if minus	$S = 1$
BV	Branch if overflow	$V = 1$
BNV	Branch if no overflow	$V = 0$
Unsigned compare conditions ($A - B$)		
BHI	Branch if higher	$A > B$
BHE	Branch if higher or equal	$A \geq B$
BLO	Branch if lower	$A < B$
BLOE	Branch if lower or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$
Signed compare conditions ($A - B$)		
BGT	Branch if greater than	$A > B$
BGE	Branch if greater or equal	$A \geq B$
BLT	Branch if less than	$A < B$
BLE	Branch if less or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$

AMIT
Beyond Education

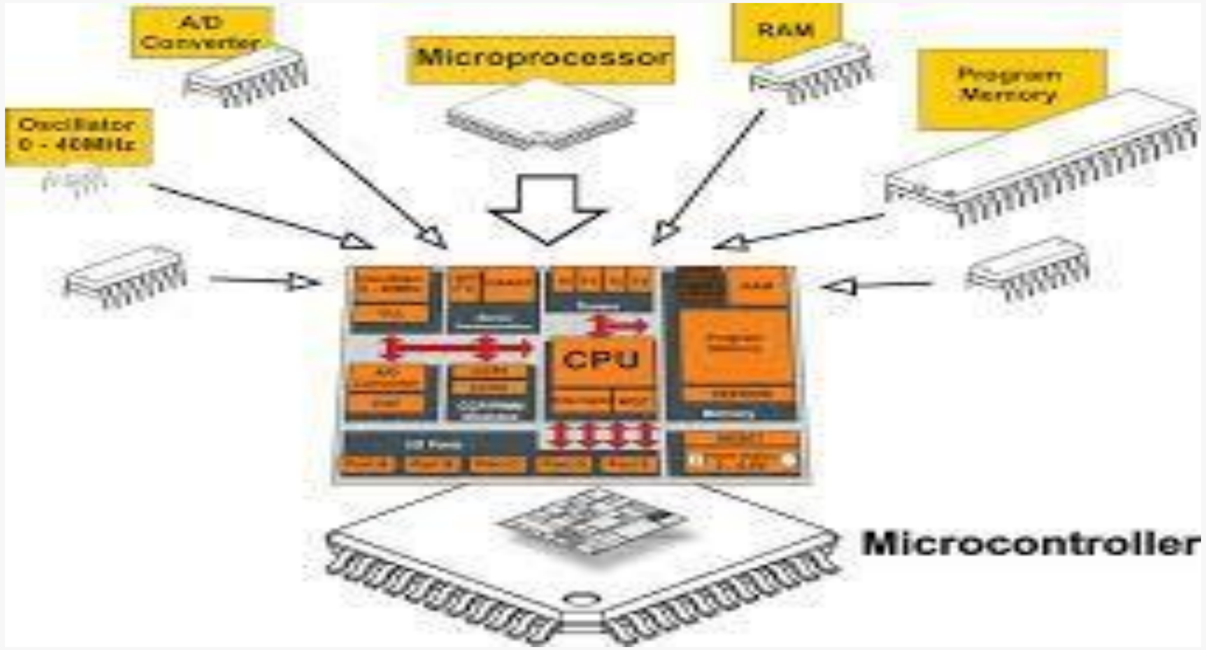
Agenda

1. Intro to digital Computer Architecture
2. Digital design review
3. Memory Types and architectures
4. Central processing unit architecture
5. Programming basic computer
6. Overview on a MCU architecture

AMIT
Beyond Education

Overview on a MCU Architecture

Difference between a microcontroller and a microprocessor



Overview on a MCU Architecture

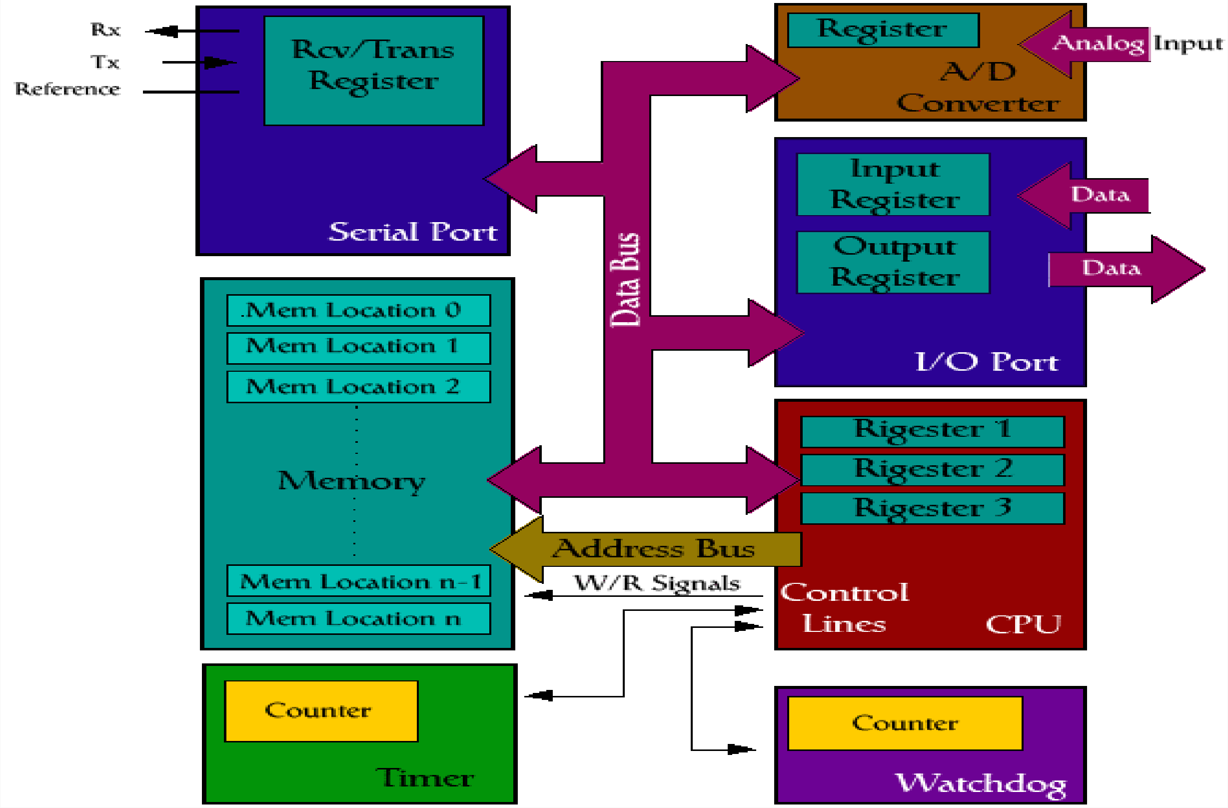
Difference between a microcontroller and a microprocessor

Microprocessors		Microcontrollers
1	It is only a general purpose computer CPU	It is a micro computer itself
2	Memory, I/O ports, timers, interrupts are not available inside the chip	All are integrated inside the microcontroller chip
3	This must have many additional digital components to perform its operation	Can function as a micro computer without any additional components.
4	Systems become bulkier and expensive.	Make the system simple, economic and compact
5	Not capable for handling Boolean functions	Handling Boolean functions
6	Higher accessing time required	Low accessing time
7	Very few pins are programmable	Most of the pins are programmable
8	Very few number of bit handling instructions	Many bit handling instructions
9	Widely Used in modern PC and laptops	widely in small control systems
E.g.	INTEL 8086,INTEL Pentium series	INTEL8051,89960,PIC16F877

AMIT
Beyond Education

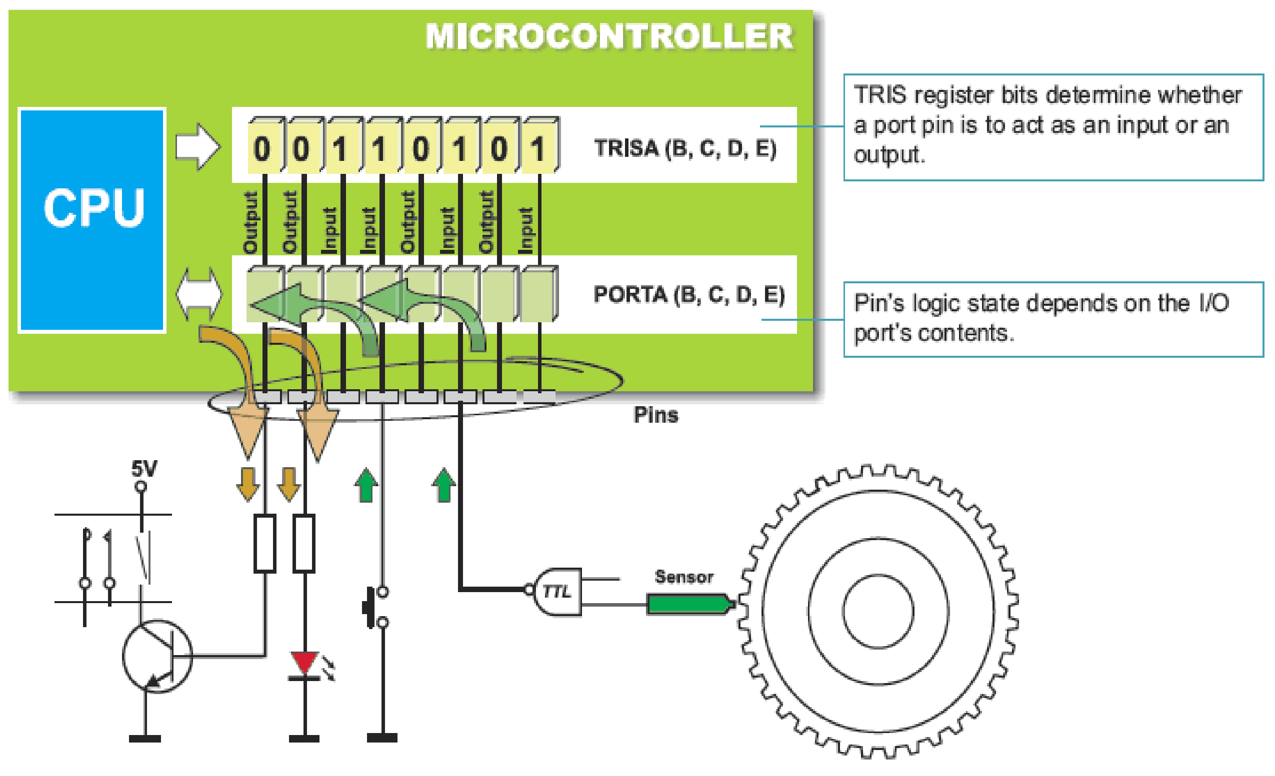
Overview on a MCU Architecture

Microcontroller internal architecture :



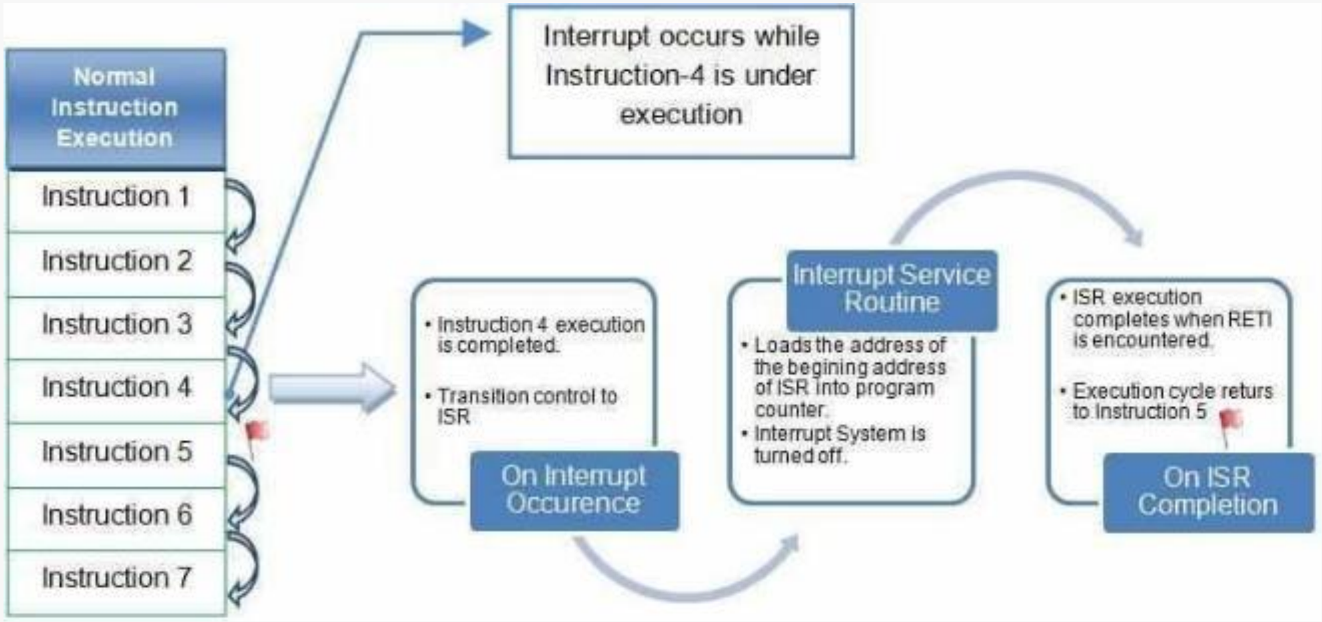
Overview on a MCU Architecture

Microcontroller internal architecture : I/O Ports



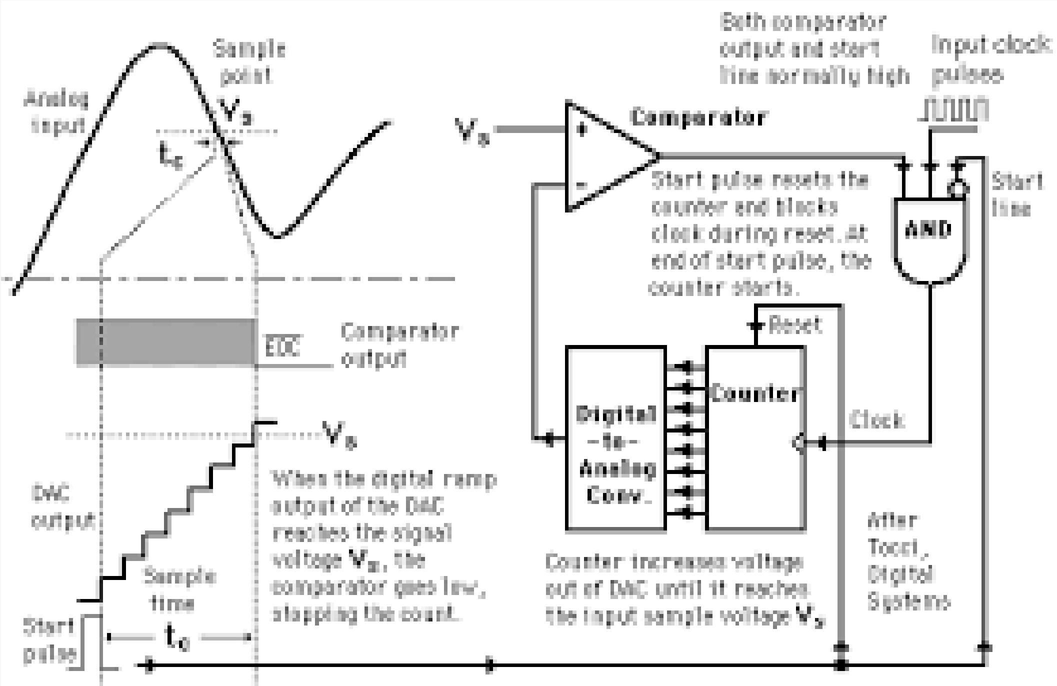
Overview on a MCU Architecture

Microcontroller internal architecture : I/O Ports



Overview on a MCU Architecture

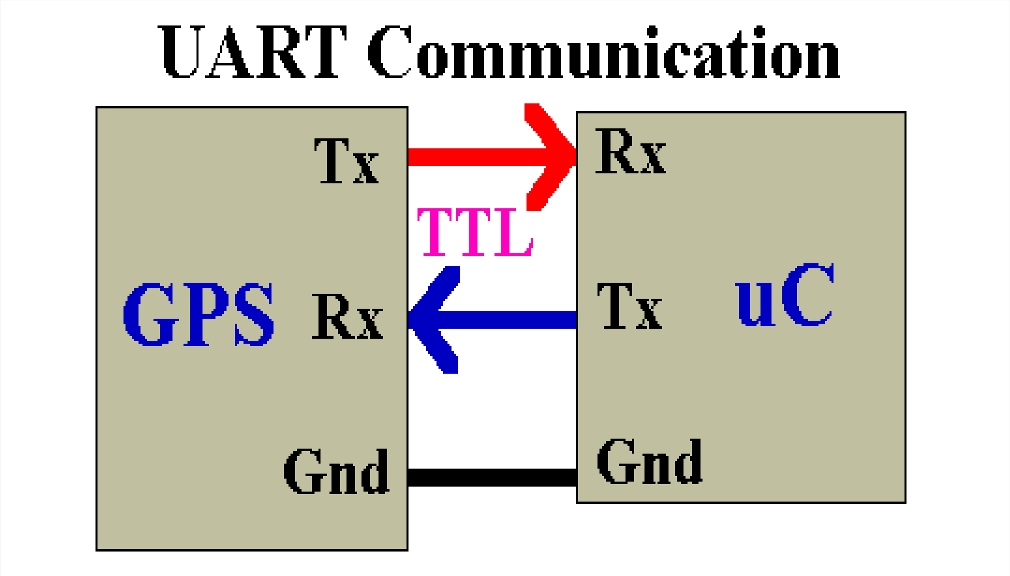
Microcontroller internal architecture : ADC



AMIT Beyond Education

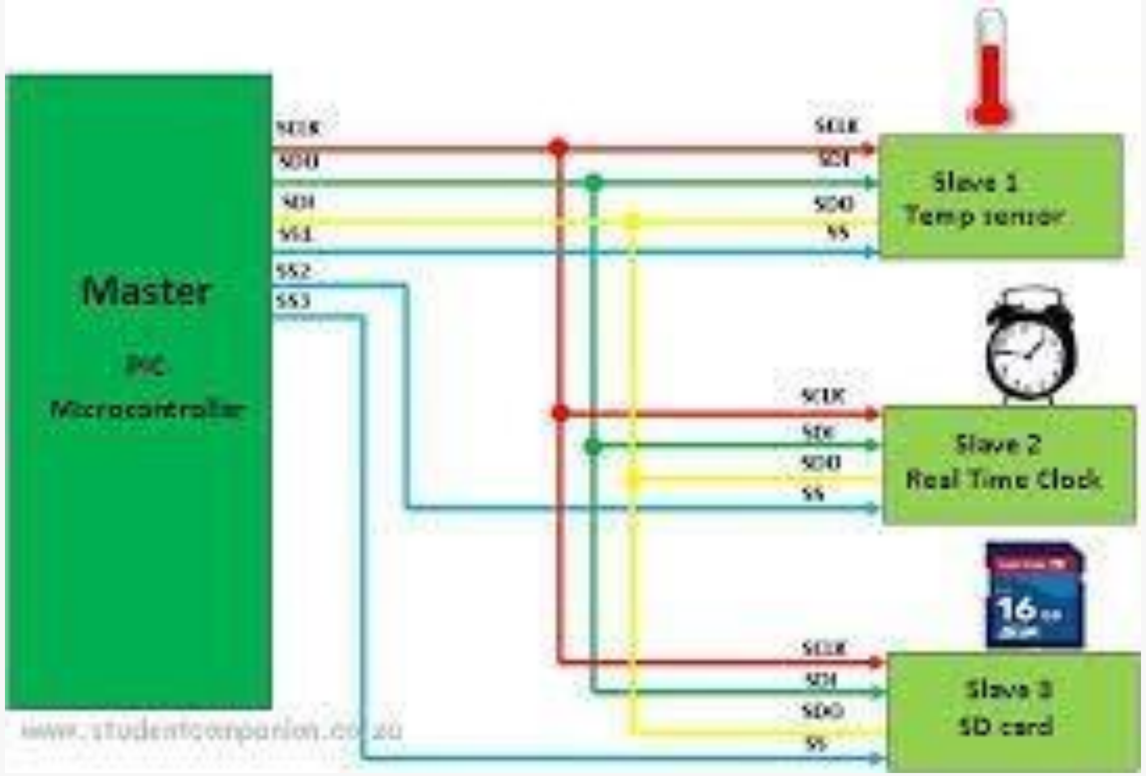
Overview on a MCU Architecture

Microcontroller internal architecture : Serial Communication



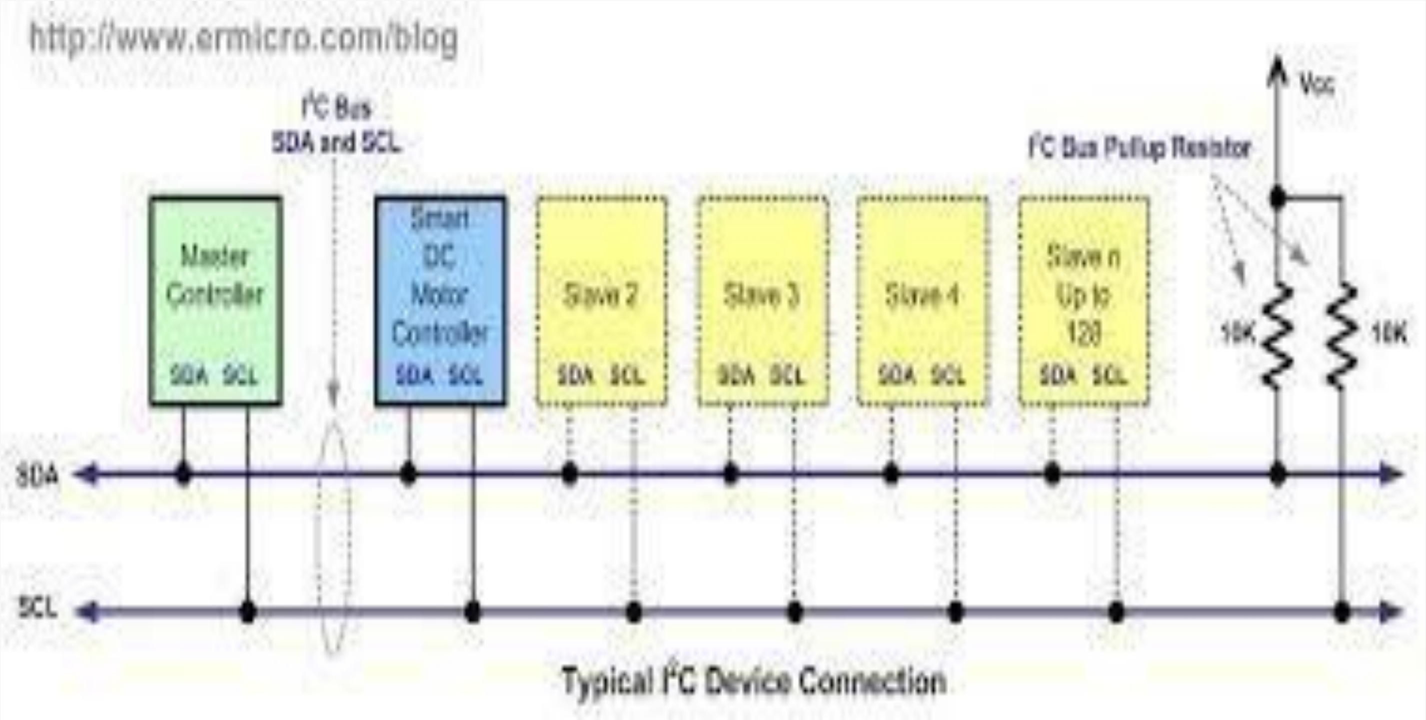
Overview on a MCU Architecture

Microcontroller internal architecture : Serial Communication



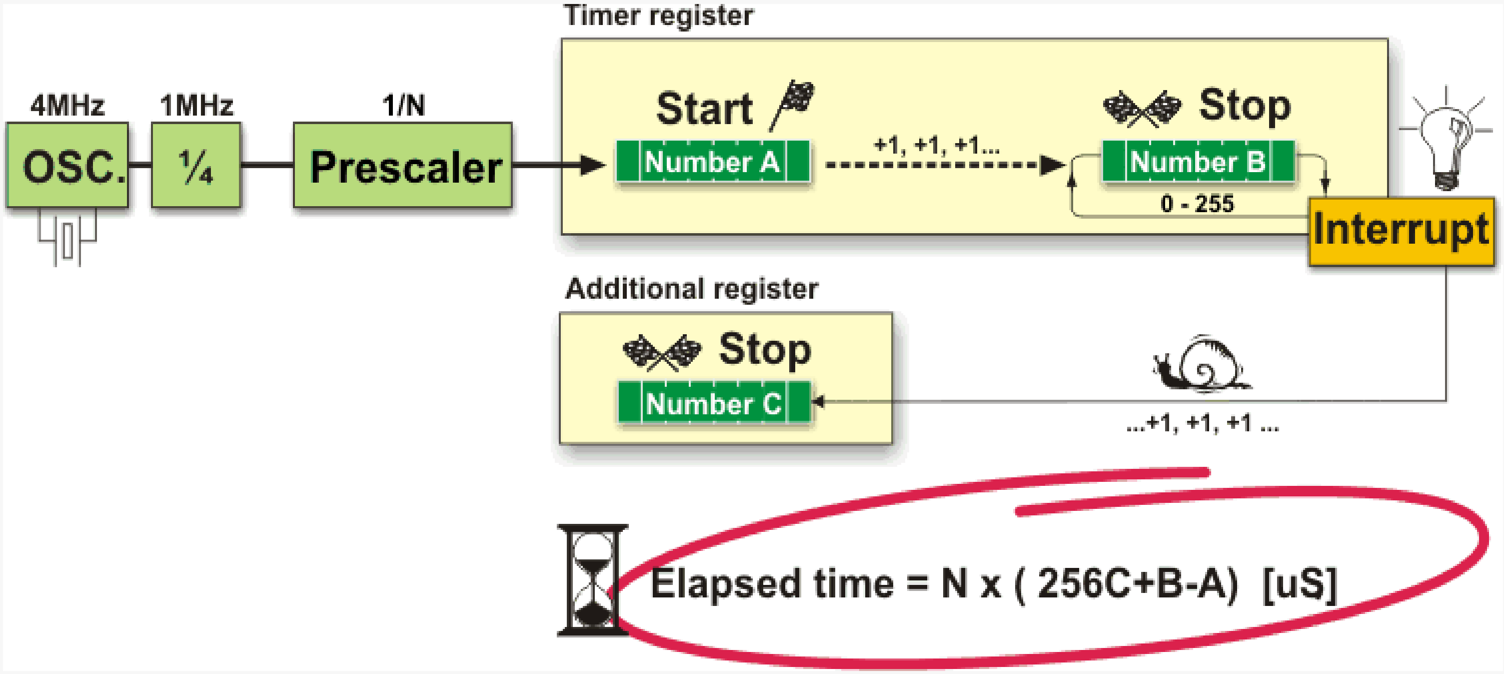
Overview on a MCU Architecture

Microcontroller internal architecture : Serial Communication



Overview on a MCU Architecture

Microcontroller internal architecture : Timer Unit



THANK YOU!

Beyond Education