Embedded Systems Concepts

Computing System Definition

A device capable for performing mathematical and logical operations in accordance with a predetermined set of instructions.

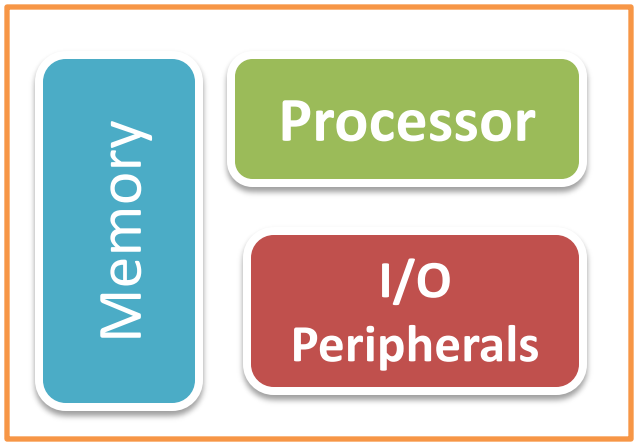
Computing System Components

Any computing system consists of 3 main components:

1- **Processor** which is responsible for performing the instructions.

2- **Memory** which is responsible for storing the program

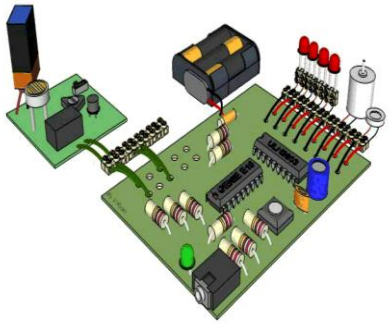
3- **Input / Output peripherals** for interacting with the user.



Embedded System Definition

Embedded system is a computing system with limited resources (Processor, Memory and I/O) used for performing a specific task.

.

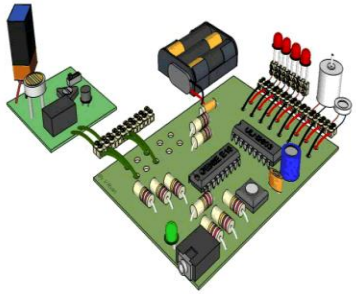


Vs.

Personal Computer

Computer is for general purpose use and with huge resources

Embedded System

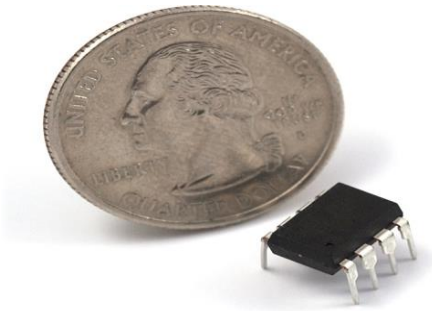


Embedded system is for specific purpose and with limited resources

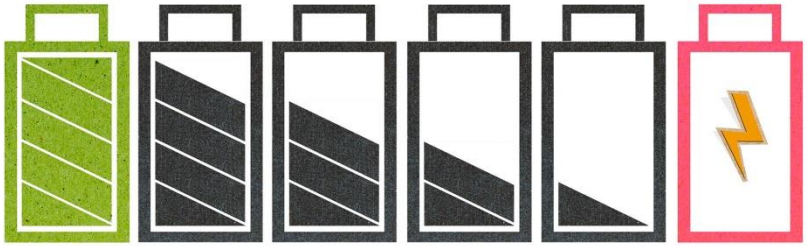
Embedded Systems Challenges

1- Performance

2- Size

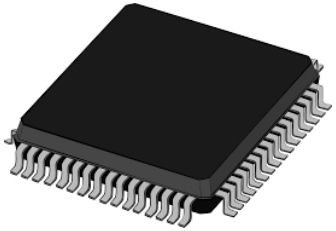


3- Power Consumption



4- Cost

ES Implementation Techniques

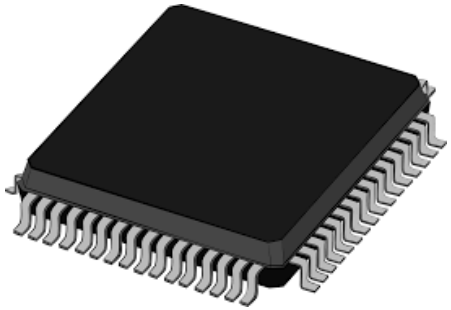
	System on Board SB	System on Chip SOC
		
performance	——	——
size	High	Low
cost	High	Low
Power consumption	High	Low
configurability	Easy	NA

When To use ... ?

System Board

For Development and Design phase

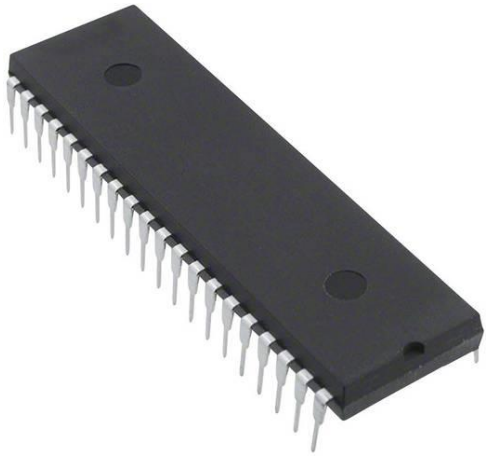
System On Chip



For Production after Design

Microcontroller Definition

The microcontroller is a system on chip consists of processor, memory and I/O peripherals. So, it looks like an IC but inside it has all the needed elements to make an ES and ready for programming.



Microcontroller Vs. Microprocessor

Microcontroller -> is a system on chip, contains processor, memory and I/O.

Microprocessor -> is one element from the needed 3 elements to make ES. So it needs memory and I/O.

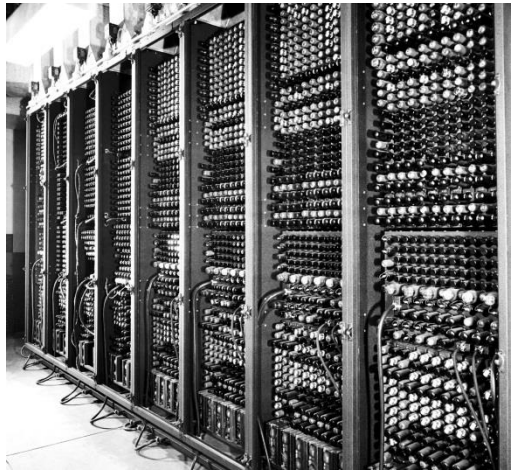
Question …

What is the difference between the following:

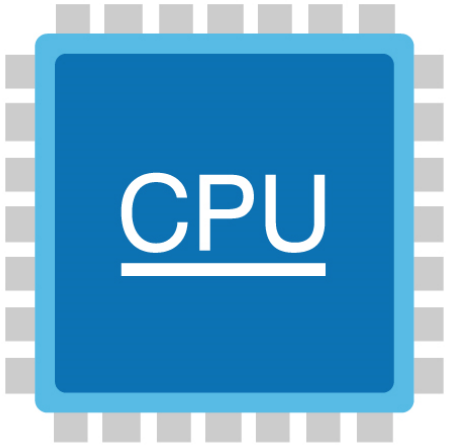
1- Processor
2- Microprocessor
3- Central Processing Unit (CPU)

Answer ...

In the past, processor was made of ***vacuum tubes***, it was very huge like a whole building. When the silicon industry came to the scene, a new processor was developed in small size. To differentiate between the two processor at this time, the vacuum tube one was called a **processor** while the silicon based IC was called a **microprocessor** (Micro refers to a smaller in size one). But in the current days, there is no processor based on vacuum tube, so there is no processor and microprocessor, all processors are small in size. So, the words processor and microprocessor are now interchangeable words. They are referring to the same thing.



When you have a system that contains more than one processor, one of them would be the master one, it would control all other processor. This one is called the **Central Processing Unit CPU**.

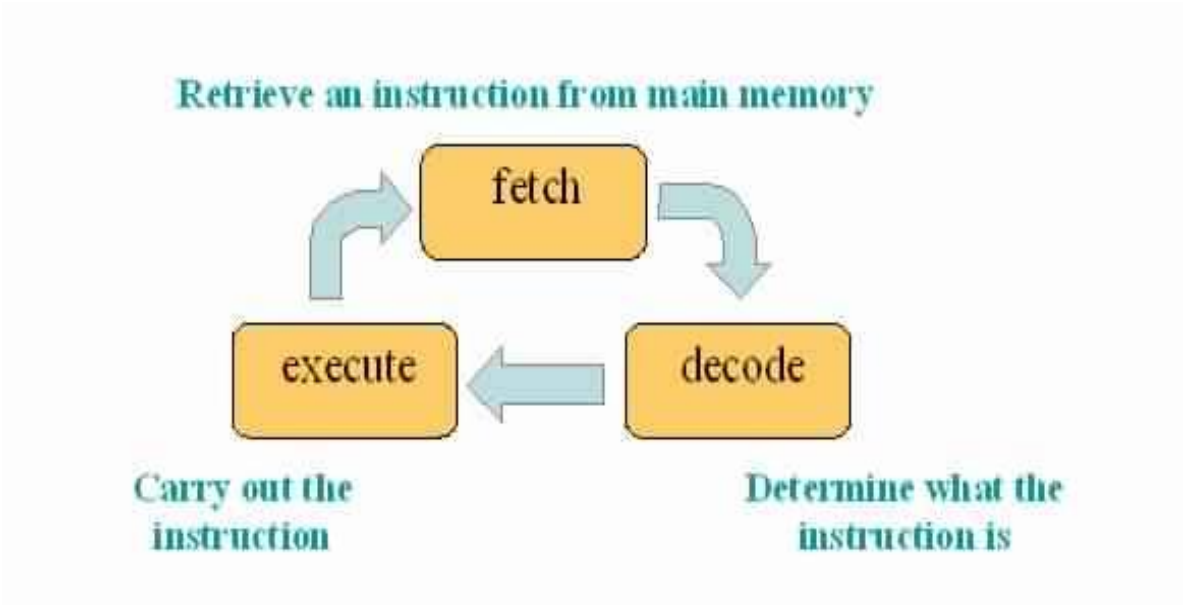


The 3 main steps by processor:

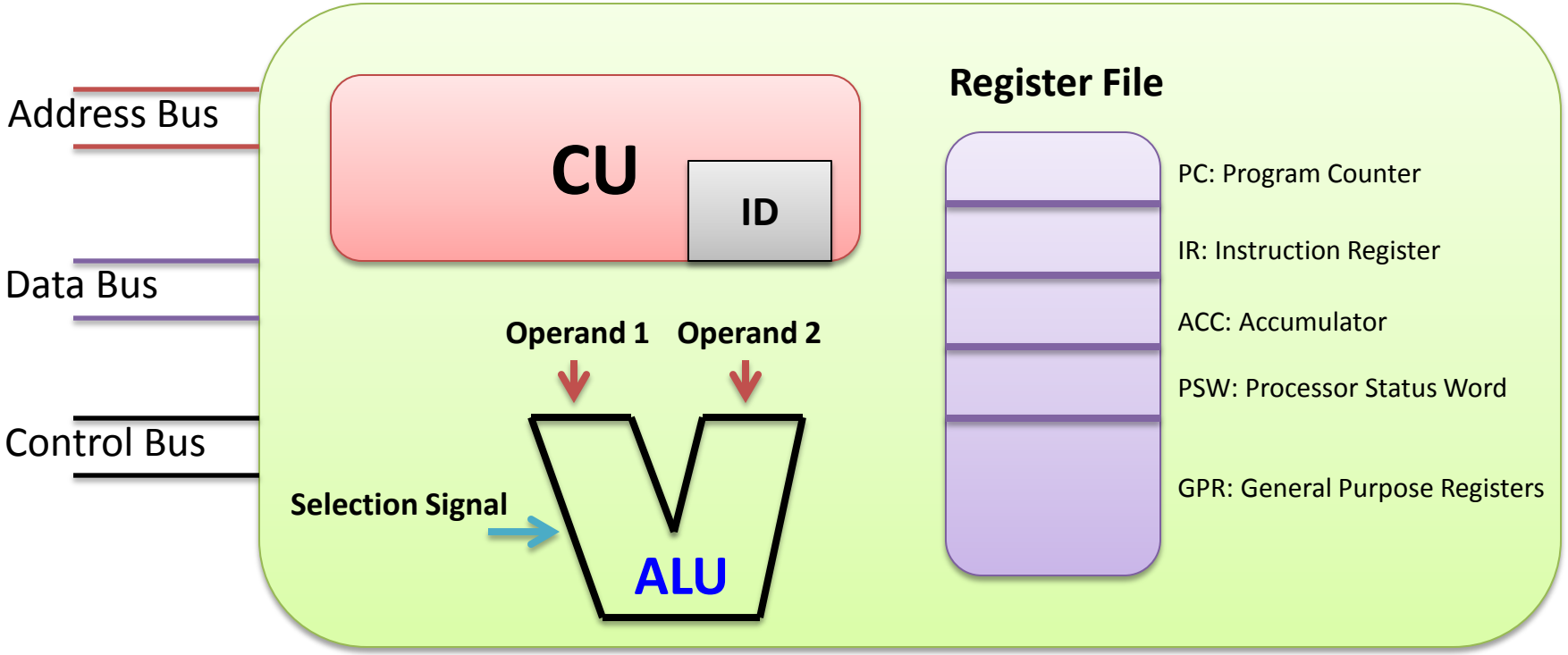
1- Fetch

2- Decode

3- Execute



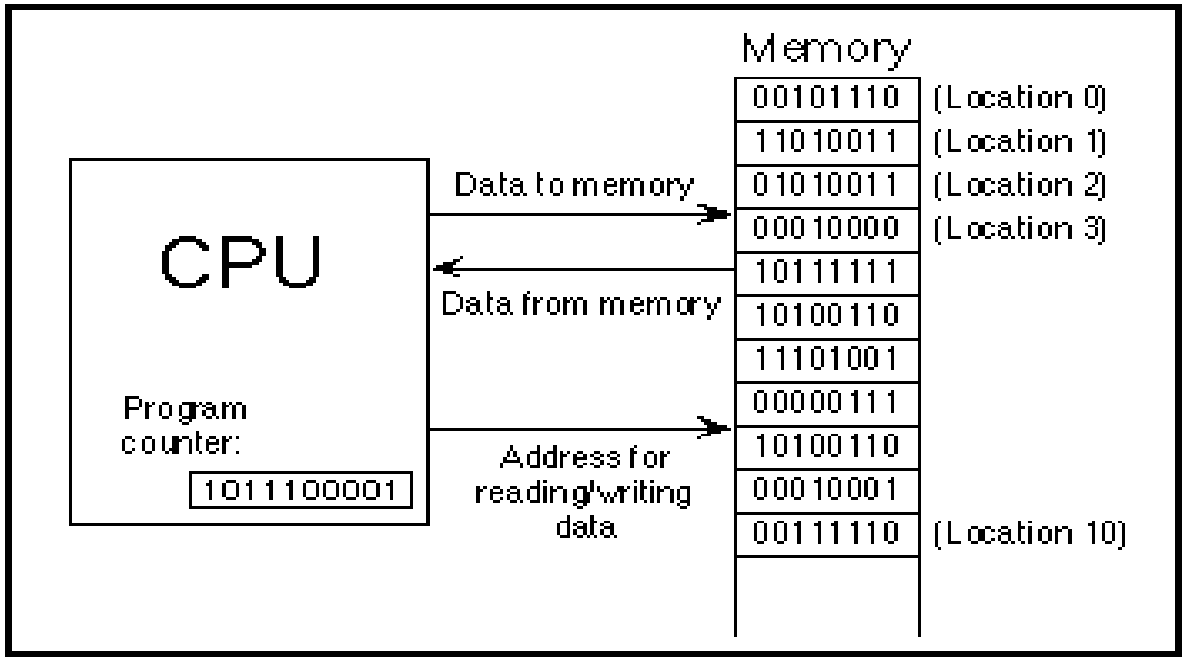
Inside the processor



CU : Control Unit
ID : Instruction Decoder
ALU: Arithmetic Logic Unit

1- Fetch :

By CU “Control Unit” from the main memory.



2- Decode :

Inside CU “Control Unit” by ID “Instruction Decoder”.

Every processor has its **instruction set** which is the group of instructions that could be executed by the processor.

Every instruction has a unique binary representation called **“Opcode”.**

The decoding step is responsible for defining the operation required by the instruction fetched from the memory, and defining the operands of this instruction then pass these information to the ALU.

Instruction Set

Instruction	Op Code
ADD	011
SUB	101
AND	110
OR	001
XOR	111
SHIFT	100

Note, The instruction set differs from a processor to other in number of instructions, type of instruction and even for op code of the similar instructions

Decoding Example

Given the shown instruction format of a certain processor and the same instruction set mentioned in the previous slide, the CU fetched the following instruction:

Instruction Format

Op Code	Operand 1	Operand 2
3 bit	3 bit	2 bit

10111011

101	110	11

Back to the instruction set, we will find that 101 is the op code for SUB operation. This means that this instruction is:

SUB 6 3

Question …

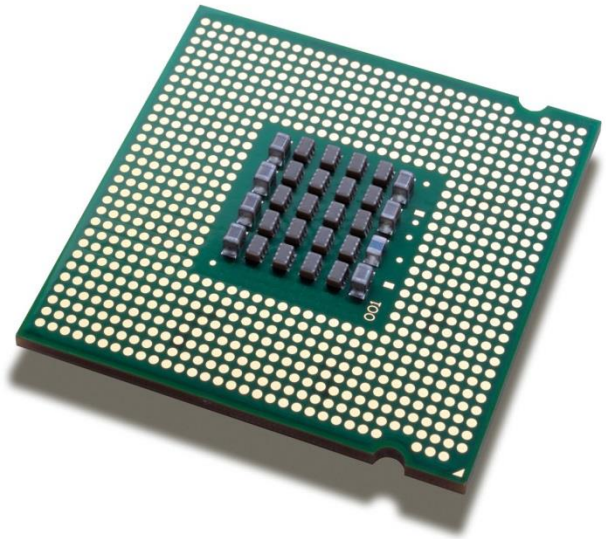
1- Is the compiler is target specific … ? I mean if a compiler for a processor X, can it compile for a processor Y … ? Why ?

2- A code is compiled for processor X, could it be executed on a processor Y … ? Why ?

Answer ...

The compiler is converting the *C source code* into *assembly* code, the assembly instructions are the set of instructions defined in the processor instruction set. The processor instruction set is *unique* per target, i.e. the instruction set differs from processor to another. For that, the compiler needs to know the instruction set for the processor which is going to compile the C source code to. For that, the **compiler is a target specific**. The compiler for AVR processor can not compile for ARM processor.

Again, for the same previous reason, the executable file is a processor specific, a code which is compiled for a processor X **can not be executed** on processor Y.



Instruction Set Architecture

ISA

	RISC	CISC
	"Reduced Instruction Set Architecture"	"Complex Instruction Set Architecture"
Performance	*Same*	*Same*
Cost	*Same*	*Same*
Size	*Same*	*Same*
Power Consumption	*Same*	*Same*

Performance

Assume the following C line of code:

```
int x = 3 * 4 ;
```

In CISC Processor

This C line of code will be converted into a **single** line of assembly, because the CISC machine includes many instructions and most probably includes the multiplication instruction.

But, The decoding phase takes many cycles in CISC machine because it decodes against a lot of instructions. It takes approximately 4 clock cycles.

In RISC Processor

This C line of code will be converted into **more than one** instruction (approximately 4) because most probably the RISC machine doesn't support the multiplication instruction.

But, the instruction in RISC machine takes only 1 clock cycle because it decodes against few numbers of instructions.

The result is that this line of code takes 4 clock cycles in both CISC and RISC machine, So they are **equivalent** in performance.

Cost

The cost of an Embedded Systems environment consists of two levels:
1- Hardware level
2- Software level (Tool chain license and development tools).

In CISC processor, the ***hardware is high cost*** because its ALU contains many logic circuits for their many instructions. But it is tool chain it not complex, as every line in the C code would be easily reflected in the instruction set thanks to its big instruction set which has a lot of options.

In RISC processor, the hardware is not high cost while the ***software development tools are very high cost*** because its instruction set is very limited to basic instructions only which makes the compiler do extra effort in order to convert the C code into assembly.

The result is that both of them will have ***the same cost*** at the end

Size

Before comparing the size of both CISC and RISC, Note that there are two types of “Instruction Decoder ID

Hardwired

Uses hardware logic gates to decode Instructions i.e. the decoding is done by algorithm implemented in hardware logic gates. It is Very fast and big in size part comparing to the next type.

Micro-Programmed

The decoding is done by software algorithm implemented on a mirco programmed memory. it is Slow, low cost and small in size.

Power Consumption

Although CISC machine has more powerful ALU, but actually only one circuit inside the ALU is enabled at time either in RISC or in CISC. For that reason CISC and RISC and consuming the **same amount of power.**

Instruction Set Architecture

So, it is now clear that CISC is not better that RISC or vice versa. It is all about the company mentality. **Intel** is one of the biggest companies in the market and it uses CISC machines, while **ARM** which is also one of the biggest companies in processor designing uses RISC.

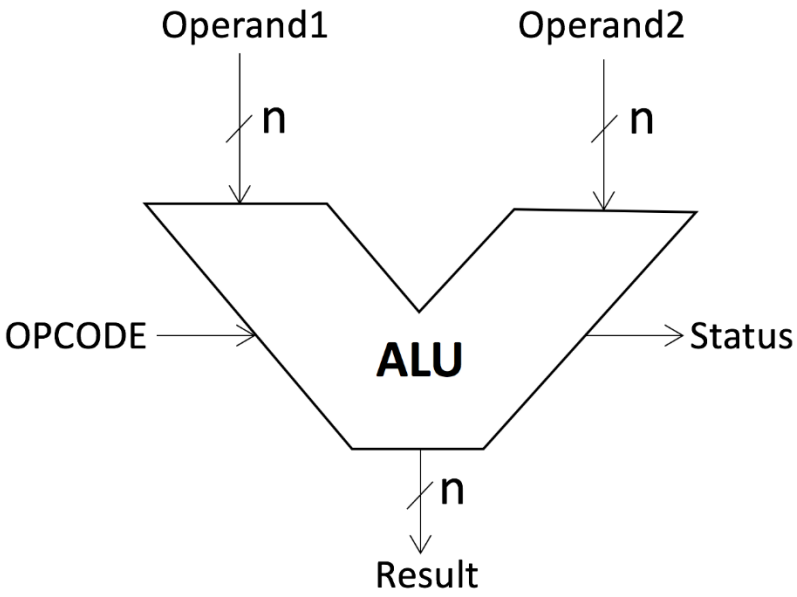
Note that not all the ISA are only CISC and RISC, other types are existing but it has some drawbacks or used for specific reasons, like:

- **OISC** "One Instruction Set Computing"
- **NISC** "No Instruction Set Computing"
- **ZISC** "Zero Instruction Set Computing"

3- Execute:

By ALU “Athematic Logic Unit”.

It contains many logic gates for all the supported instructions.



Register File

It is certain type of memory existing in the processor. Each word is called register and has a specific usage. The register file differs from processor to other, but the following are the common registers:

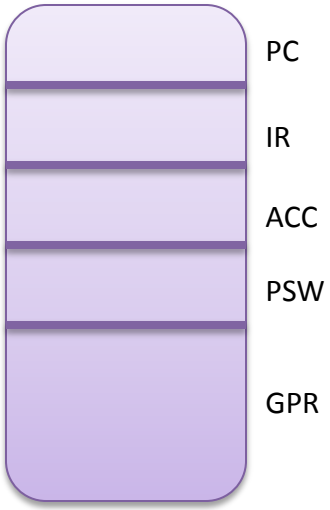
PC "Program Counter" : Address of the next instruction to fetch, decode and execute.

IR "Instruction Register": Fetched instruction to be decoded and executed.

ACC "Accumulator" : The Real Result From ALU.

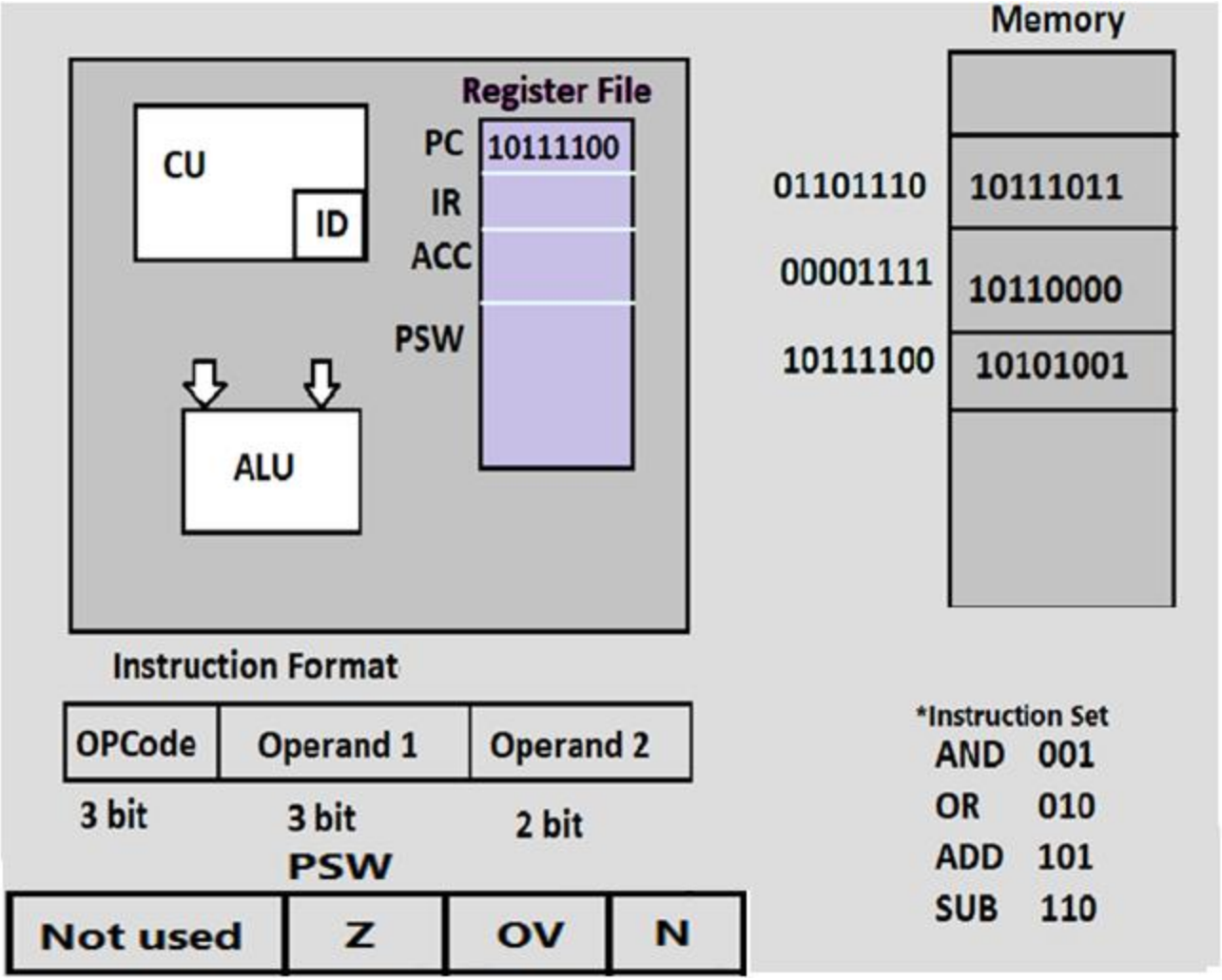
PSW "Processor Status Word": Word Byte Holds flags about the result "Status".

GPR "General Purpose Registers": General registers used by the user to define high speed accessed data using the C keyword 'register'.

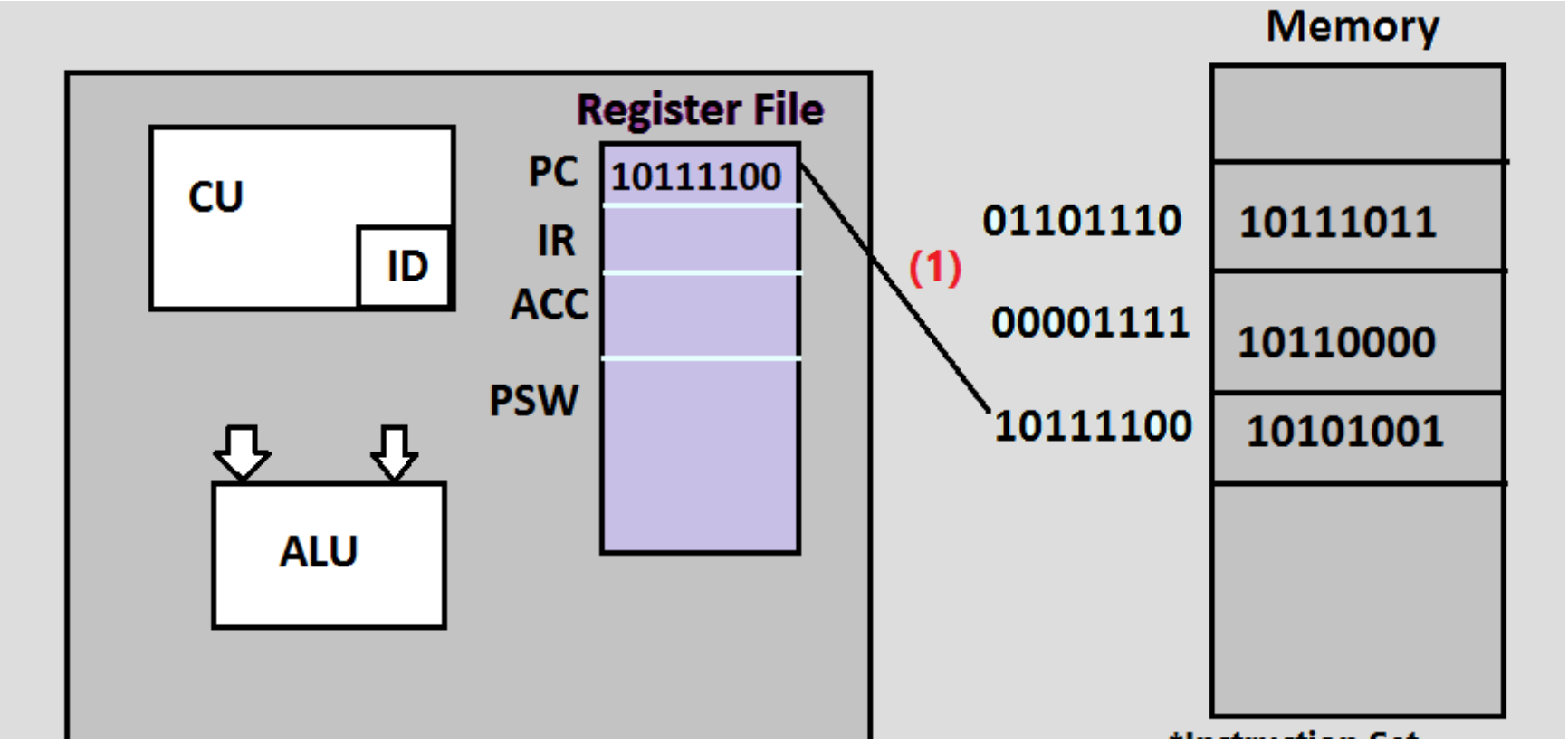


Exercise

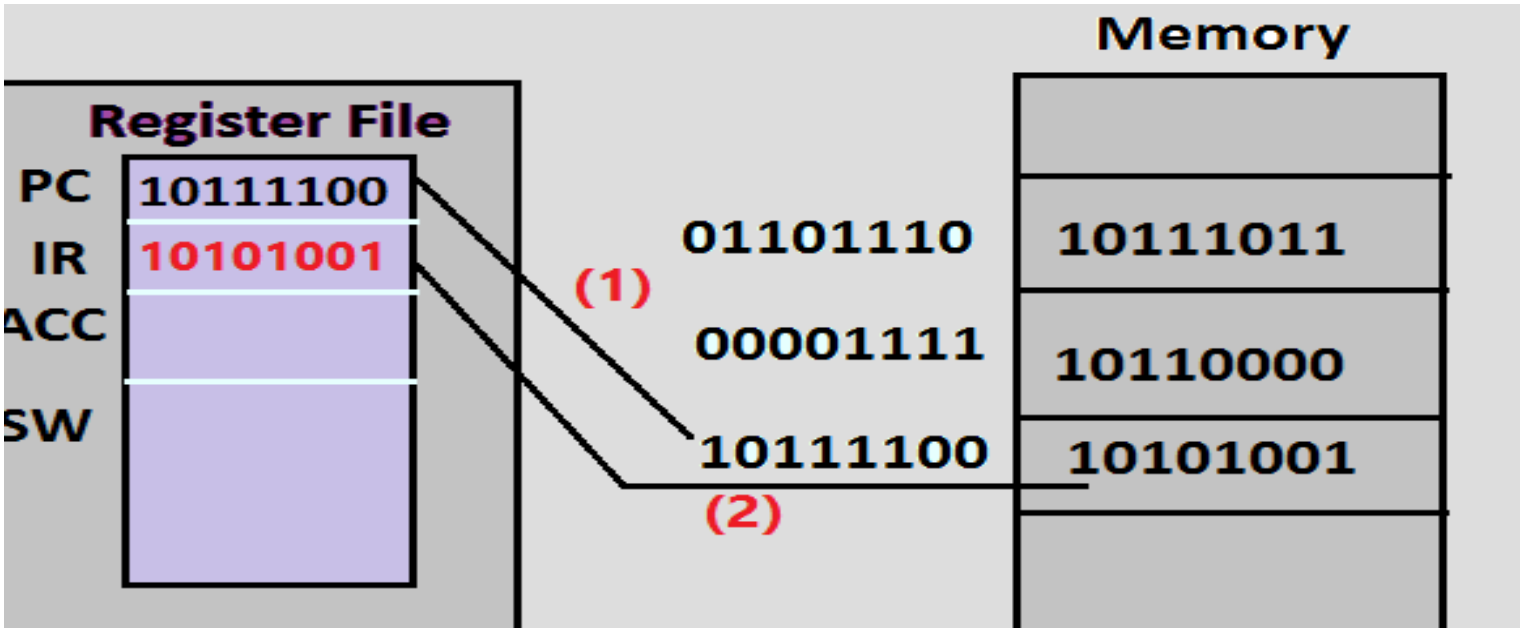
Consider the following processor, perform one complete processor cycle



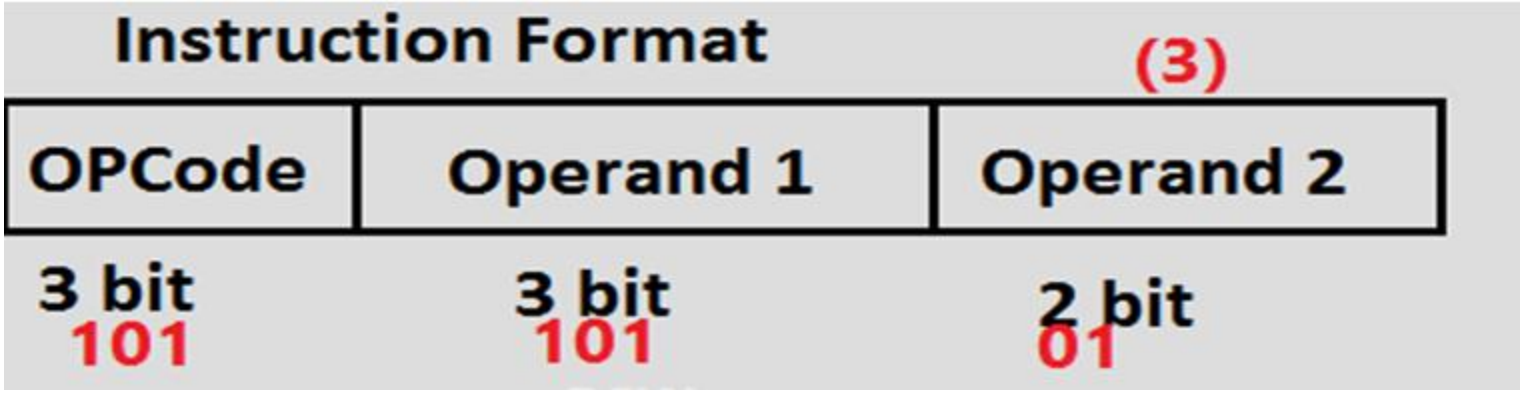
Step1: Fetch

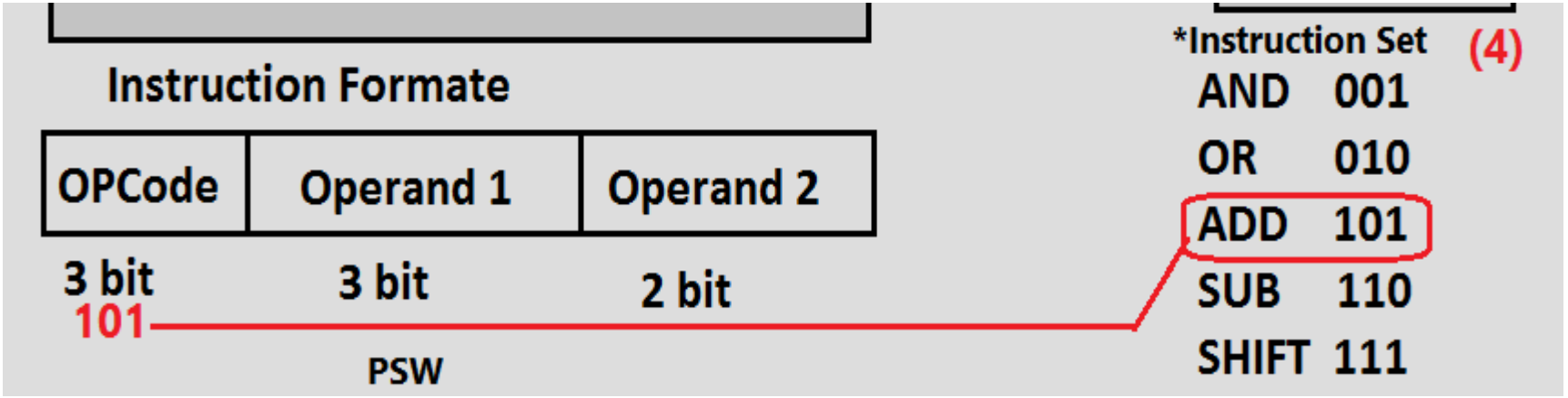

Memory
Register File
CU
ID
PC
10111100
IR
ACC
PSW
ALU
(1)
01101110
00001111
10111100
10111011
10110000
10101001

Step1: Fetch

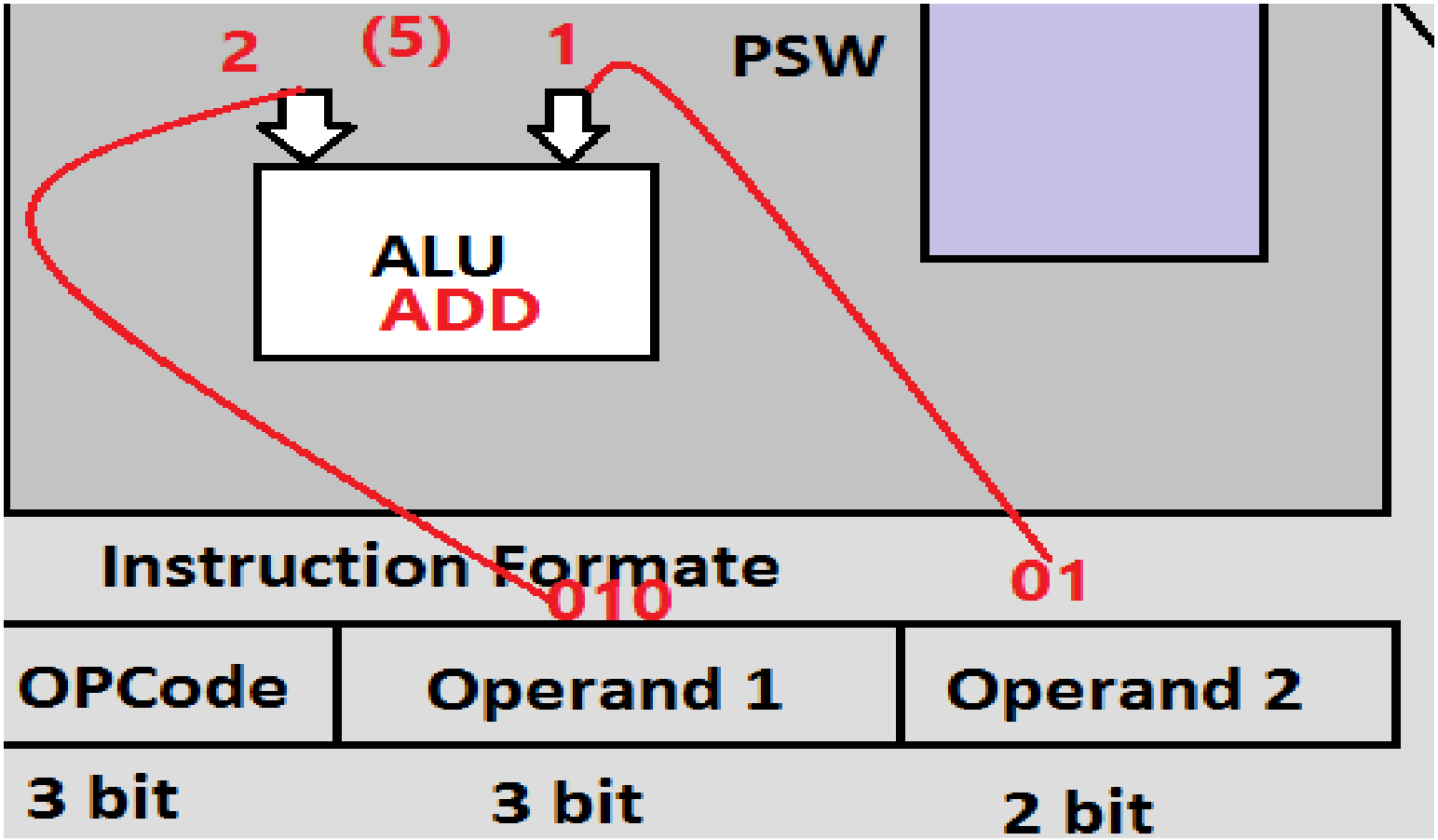

Memory
Register File
PC
10111100
IR
10101001
ACC
SW
(1)
01101110
00001111
10111100
(2)
10111011
10110000
10101001

Step2: Decode

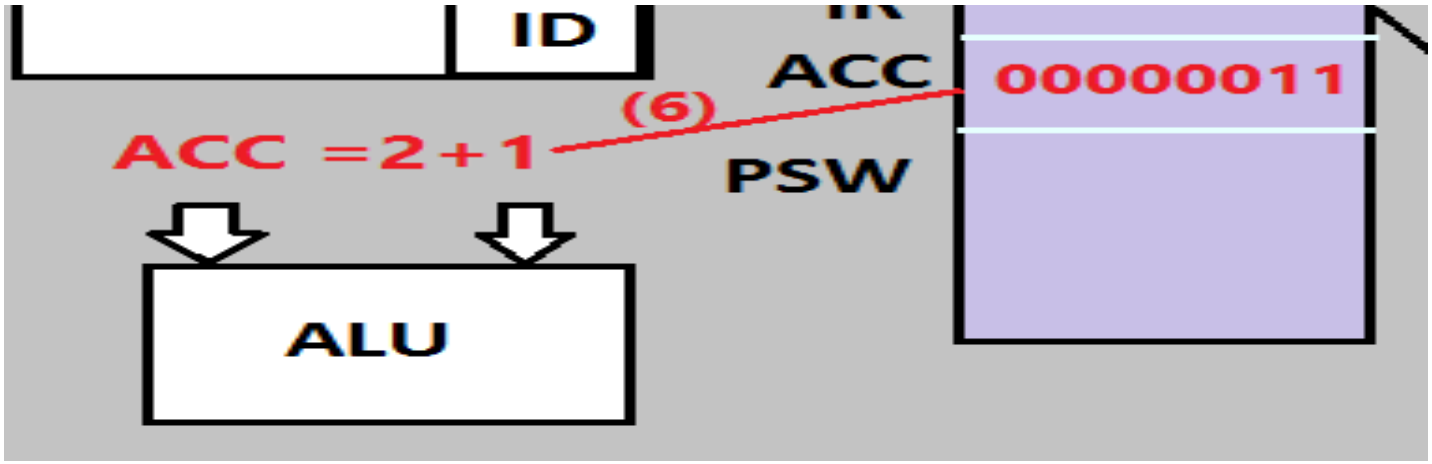




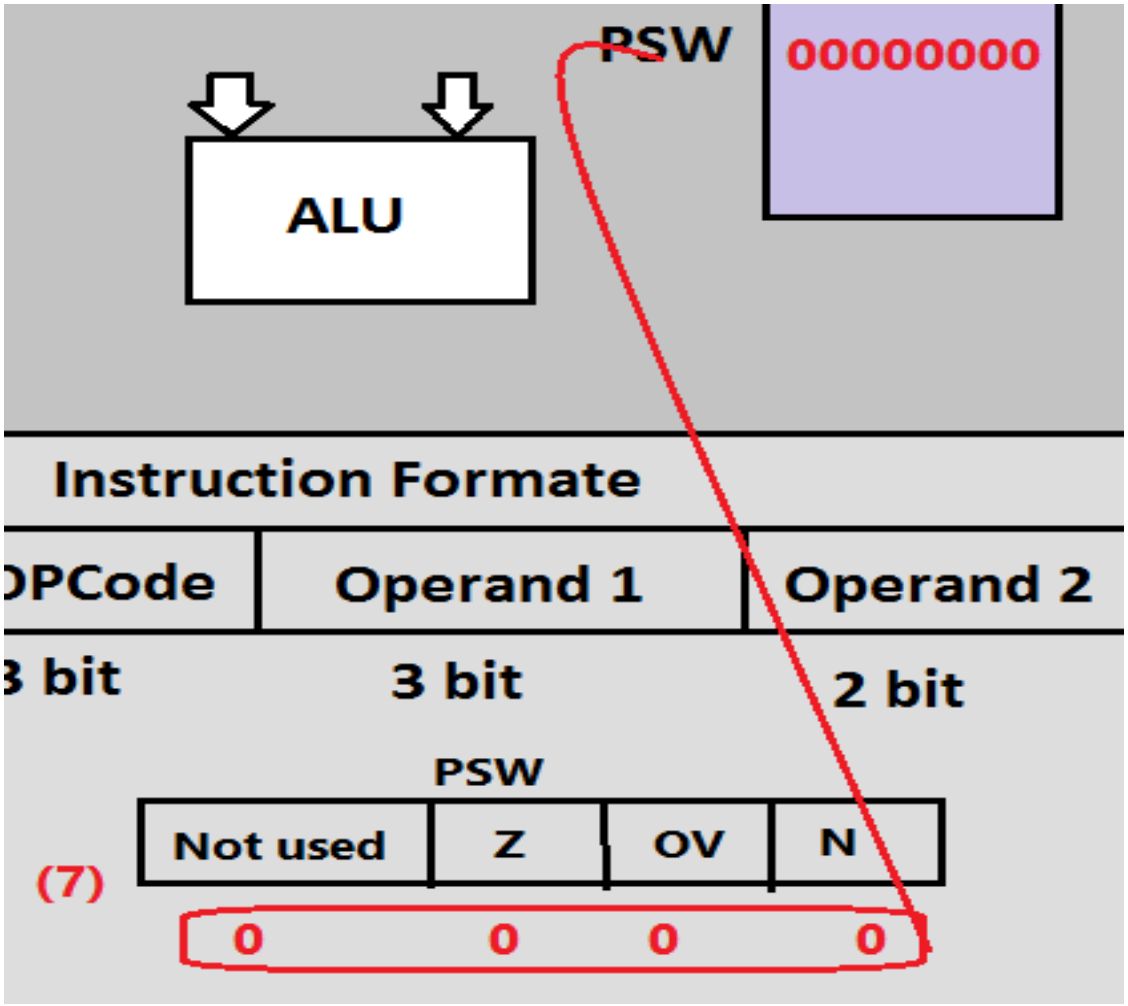
Step2: Decode



Step3: Execute



Step3: Execute



The End …

IMT SCHOOL

www.imtschool.com

ww.facebook.com/imaketechnologyschool/