ZUFS - Task 1

What is an embedded system?

Ans: it is a computing system that has limited resources because it is used for a specific function, an the word “embedded” refers to the system being in most cases a small system embedded in another system for a specific task to control and maintain the system.

EX: air conditioner and other home appliances, projector

For a computing system to be functional => 3 components are required:

1. Processor: the part that executes instructions
2. Memory: it stores the data
3. I/O peripherals

Ways to implement an embedded system:

1. System on board (SOB): **i.e.** when you get the IC for the processor, IC for the memory, IC for I/O peripherals, and put them on a board that is in most cases big enough to fit in a hand
2. System on chip (SOC): **i.e.** to call the manufacturer to implement the whole system on a chip and fabricate it.

Embedded system challenges:

1. Performance
2. Size
3. Cost
4. Power consumption

   For all those points, the SOC wins (except in performance) but what makes the SOB still used to this day is its **Configurability**, as any system must go through an R&D stage before going to mass production.

Microcontroller definition: is a complete system (processor, memory, and i/o peripherals)

Microprocessor definition: is only one of the system components

Difference between Processor, microprocessor, and CPU:

The processor: was based on the old vacuum tubes.

Microprocessor: was based on transistors.

CPU: is more of a communication controller between different microprocessors.

A processor has these crucial 3 parts:

1. CU (control unit)
2. ALU (arithmetic logic unit)
3. Register File

What the processor does when executing code:

1. Fetching: The Control Unit fetches the data from the memory
2. Decoding: the Instruction Decoder (a part inside the CU) decodes instructions based on the instructing format and instruction set for different operations
3. Executing: the ALU executes the code and does arithmetic (add, multiply, etc.) and logic (AND, OR, etc.) operations

ISA (instruction set architecture):

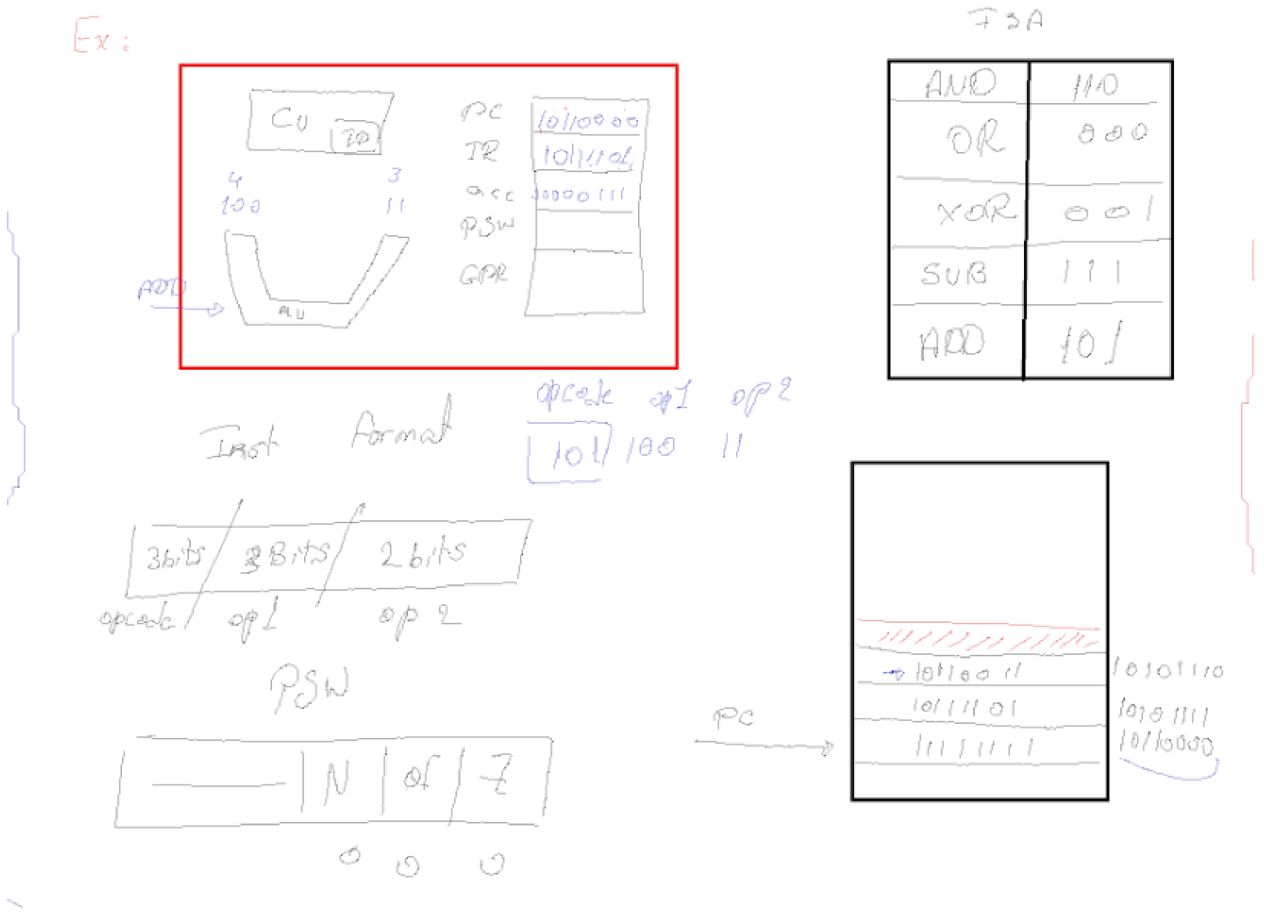
1. RISC (reduced instruction set computing)
2. CISC (complex instruction set computing)

In embedded system challenges, they are on par with each other as in performance as for simplicity the RISC tend to do more of the same operation to do one CISC operation, cost as for cheap hardware an expensive tool chain software is used to decode complex operation and vice versa for the CISC, for size is quite the same as the ALU is in fact smaller in the Risc but the ID is hardwired which makes it bigger, but for the Cisc it is microprogrammed so it is smaller, for power consumption it is pretty similar too.

Register Files:

1. PC (program counter): points to the next instruction to be executed
2. IR (instruction register): points to the instruction to be executed now
3. ACC (accumulator register): hold the result of the ALU
4. GPR (general purpose register)
5. PSW (processor status word): holds the flagis, i.e. zero flag, negative flag, overflow flag
6. SP (stack pointer)

and an example for how instructions are executed in a processor:



Memory:

1. Non volatile
2. Volatile:
    1. DRAM (dynamic RAM) which is based on capacitors
    2. SRAM (static RAM) which is based on transistors

SRAM is preferred for embedded systems as the capacitors need Refreshing Circuits which is overly consuming for the limited resources the system has, and while the SRAM IS more expensive, mostly minimal amounts of memory are required.

Explaining Video Link for Task 1

ZUFS - Task 2

Memory:

1. Volatile
2. Non-volatile

   1. NVRAM:

   A) By hardware using SRAM and a Battery to refresh the data, *its drawbacks*: non-durable, extra cost, extra size
   B) By software using EEPROM (emulated) which consists of the RAM, a flash, and a circuit called ”brown out reset”

   2. ROM:

   A) Masked ROM
   B) OTP ROM
   C) EPROM
   D) EEPROM
   E) Flash ROM

   Flash vs EEPROM:
   for the performance they are similar, when Accessing the EEPROM, it has to be accessed one byte at a time while the flash memory is accessed block at a time, for Cost the EEPROM is higher than flash because of its accessing mechanic, for Endurance(or how many a ROM have Erase/write cycle) the EEPROM has a higher number of cycles than flash, the flash memory is preferred in embedded system.

Any microcontroller **MUST** have Flash and RAM, while EEPROM isn’t mandatory

Connection between components:

1. With wire
2. On a PCP which then the connection is called *track*
3. On an IC which then the connection is called *tunnel*

What is a bus?

It is a group of connections, and there is mainly 3 buses for the processor and memory:

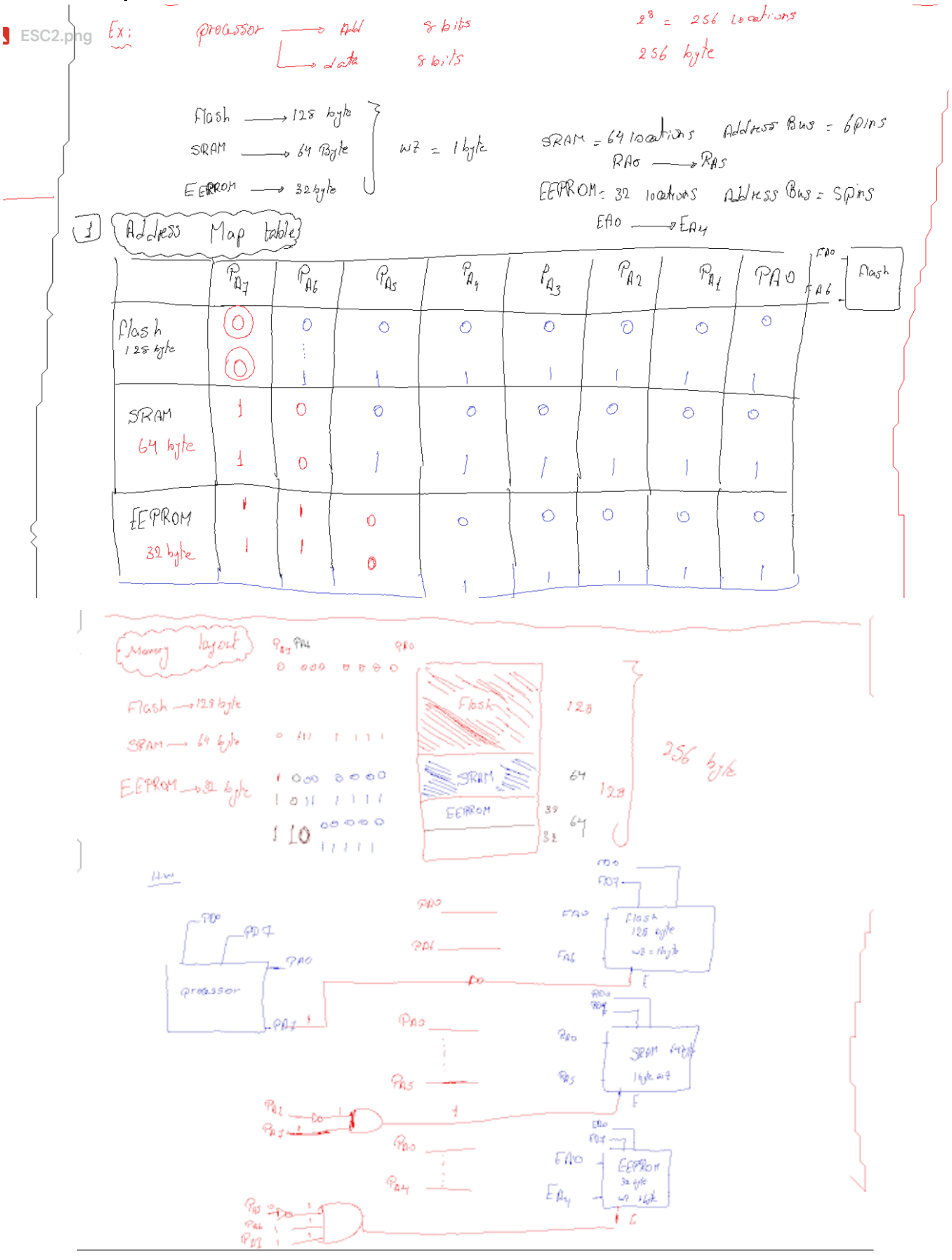
1. Adress bus (which depends on how many bits the memory has and the word size of the data, i.e. for 1024 byte memory and 2 byte word, there are a total of 512 memory locations so then the bus needs to be 9-bit)
2. Data bus
3. Control bus

Ways to connect the processor to the memory?

1. H.W. connection
2. Adress map table
3. Memory layout

An example:

ESC2.png



Ex: processor → Add 8 bits
→ data 8 bits

$2^8 = 256$ locations
256 byte

Flash → 128 byte
SRAM → 64 Byte
EEPROM → 32 byte
} wz = 1 byte

SRAM = 64 locations Address Bus = 6 pins
RA0 → RA5

EEPROM = 32 locations Address Bus = 5 pins
EA0 → EA4

1 Address Map table

	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Flash 128 byte	0	0	0	0	0	0	0	0
	0	1	1	1	1	1	1	1
SRAM 64 byte	1	0	0	0	0	0	0	0
	1	0	1	1	1	1	1	1
EEPROM 32 byte	1	1	0	0	0	0	0	0
	1	1	0	1	1	1	1	1

Memory layout

Flash → 128 byte
SRAM → 64 byte
EEPROM → 32 byte

ZUFS - Task 3

Fragmentation: is when there is unused space in the memory

Homogenous: when the memory is after each other, not separated like nodes in linked lists i.e.

Special cases:

When a processor's data bus size is larger than the memory word size, A loss in data happens.

When the opposite happens, data loss can occur and there is unused memory as well.

When a processor's data space it can see is larger than the no. of locations, it then must access what is lower than its potential

But when the opposite happens, then a space of memory can't be accessed which then a component named:
MMU (memory management unit) is needed which is a part that makes the processor access more memory using paging technique, but for using an MMU additional time to communicate with the MMU is lost and some memory locations can't be accessed like the paging register i.e.

Digital design:

Van Newman vs Harvard architecture
Van Newman architecture makes the processor fetch data, decode instructions, and then execute it, while each process takes its time the other parts are just off or not working.

Harvard architecture makes use of the parts when its job is done so when the fetching circuit has done its job and the instruction is sent to be decoded, it then fetches other data and when the decoder finishes decoding the old instruction the new instruction is decoded and so on, so that more than one instruction is executed in one cycle.

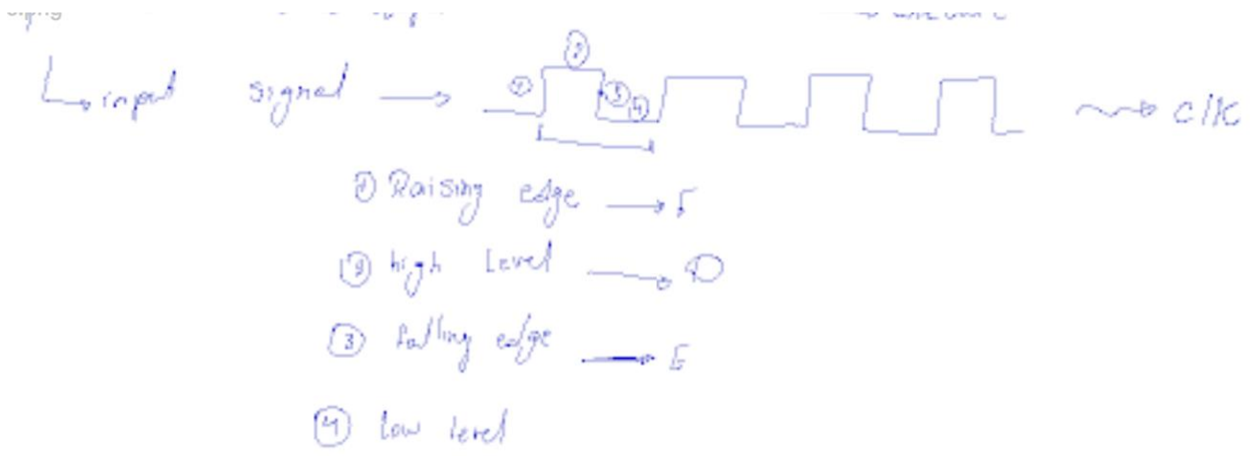
But of course, the increase in speed for Harvard architecture comes with an increase in complexity to write the necessary code.

In AVR architecture there is 2 main ways to map the buses to the I/O memory:

1. Port mapping: which is direct bus from the processor to the memory
2. Memory mapping: which is using the same bus of the SRAM to map also to the I/O memory

   Differences is that port mapping is faster but more complex, while memory mapping is slower but simpler as it is the same address being transferred.

When we go back to the processor’s processes and dive deeper into the signal the processer inputs(being digital signal or square signal) the fetching, decoding, and executing happens at specific moments of the signal(fetching at the rising edge, decoding at the high level, executing at falling edge i.e.)

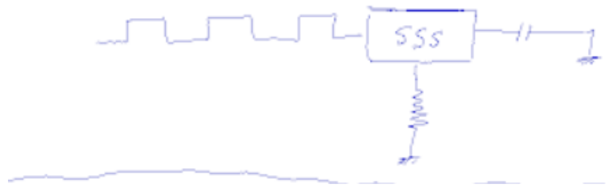


MIPS: million instructions per second
and is equal to (CLK/ no. of cycles the instruction needs)

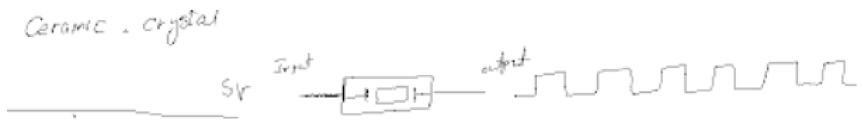
CLK: is the frequency of the system
or is 1/(the time it takes to do one signal period)

Ways to make a CLK source:

1. Electrical: using the RC system



2. Mechanical: using either ceramic or crystal to make vibrations



Comparison points	RC	Crystal	Ceramic
Accuracy	worst	best	Medium
Cost	lowest	highest	Medium
Settling time	highest	lowest	Medium
Temperature noise immunity	Worst	Best	Best
Vibration noise immunity	Best	worst	Worst
EMI(electro.....) noise immunity	worst	Best	best