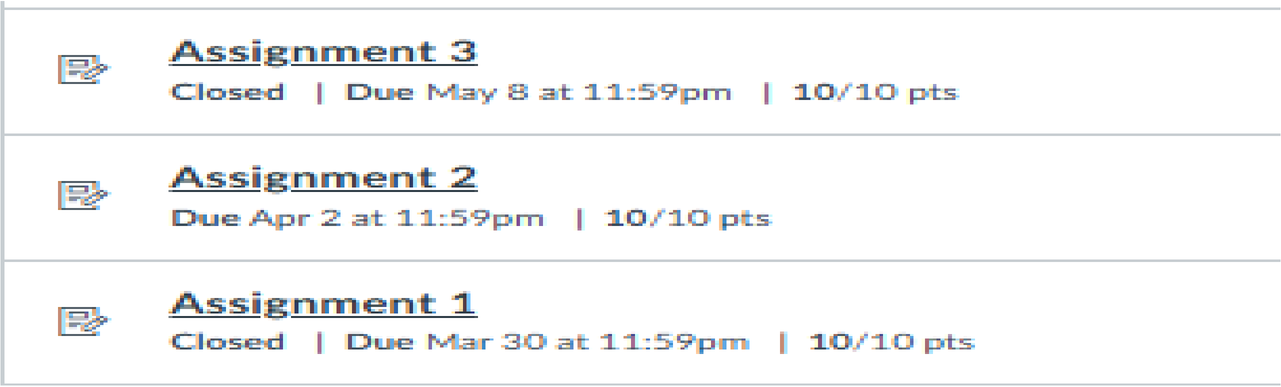
Marina Reda Abdullah 221101235 AIS

Assignments



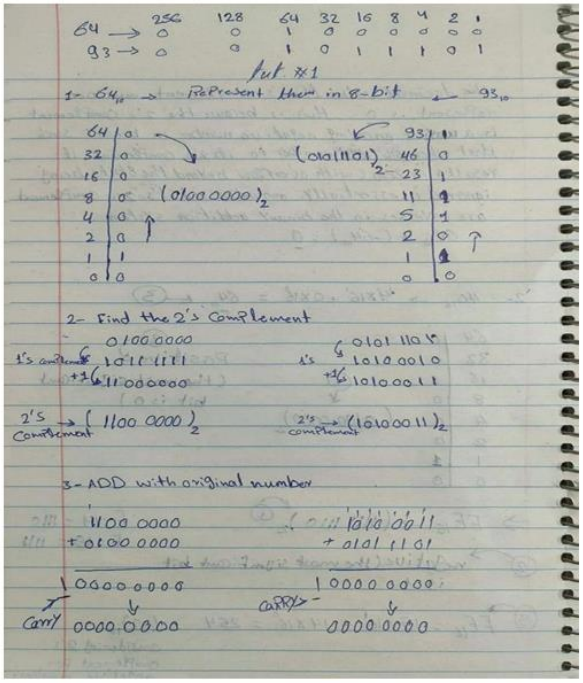
Assignment 3
Closed | Due May 8 at 11:59pm | 10/10 pts

Assignment 2
Due Apr 2 at 11:59pm | 10/10 pts

Assignment 1
Closed | Due Mar 30 at 11:59pm | 10/10 pts

CSE272 EMBEDDED SYSTEMS

Tut. # 1



	256	128	64	32	16	8	4	2	1
64 →	0	0	1	0	0	0	0	0	0
93 →	0	0	1	0	1	1	1	0	1

Tut #1

1- 64_{10} → Represent them in 8-bit ← 93_{10}

64	0
32	0
16	0
8	0
4	0
2	0
1	1
0	0

$(01000000)_2$

$(01011101)_2$

93	1
46	0
23	1
11	1
5	1
2	0
1	1
0	0

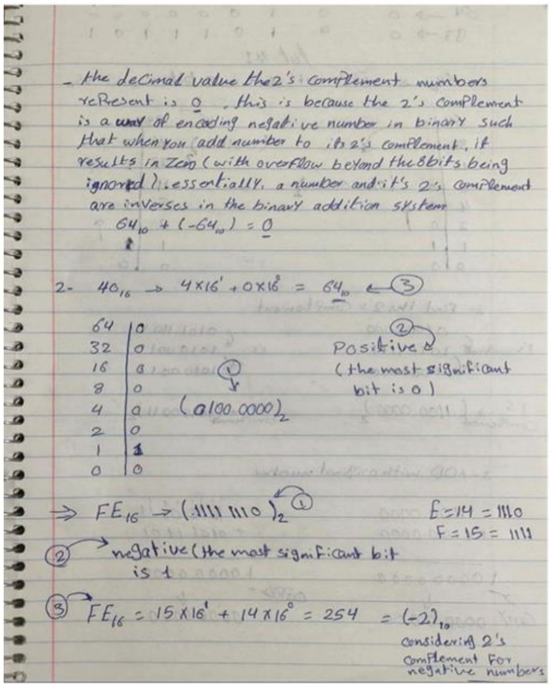
2- Find the 2's Complement

```
              0100 0000                      0101 1101
1's complement 1011 1111              1's    1010 0010
          +1  1100 0000               +1     1010 0011

2's complement → (1100 0000)₂         2's complement → (1010 0011)₂
```

3- ADD with original number

```
      1100 0000                  1010 0011
    + 0100 0000                + 0101 1101
    -----------                -----------
     1 0000 0000                1 0000 0000
carry  ↓                    carry    ↓
      0000 0000                  0000 0000
```



- the decimal value the 2's complement numbers represent is 0. this is because the 2's complement is a way of encoding negative number in binary such that when you add number to its 2's complement, it results in zero (with overflow beyond the 8bits being ignored). essentially, a number and it's 2's complement are inverses in the binary addition system

$64_{10} + (-64_{10}) = \underline{0}$

2- $40_{16} \rightarrow 4 \times 16^1 + 0 \times 16^0 = \underline{64_{10}} \leftarrow$ (3)

64	0
32	0
16	0
8	0
4	0
2	0
1	1
0	0

(1) $(0100\ 0000)_2$

(2) Positive (the most significant bit is 0)

⇒ $FE_{16} \rightarrow (1111\ 1110)_2$ (1)

E = 14 = 1110
F = 15 = 1111

(2) → negative (the most significant bit is 1

(3) → $FE_{16} = 15 \times 16^1 + 14 \times 16^0 = 254 = (-2)_{10}$ considering 2's complement for negative numbers

Marina Reda Abdullah 221101235 AIS

Microprocessor Based Systems and Microcontrollers

3. A microprocessor is said to be 16-bit. Can we specify the size of its address and data buses and registers?

Data Bus: 16 bits wide, enabling the microprocessor to handle 16 bits of data at a time.

Address Bus: The width can vary. A wider address bus allows accessing a larger memory space. For instance, a 20-bit address bus can access 1 MB of memory.

Registers: Typically match the data bus width for efficient processing, so they would also be 16 bits.

4.

- The capacity of a memory block is specified as 32K*16 bits. Find out the address and data lines for the block.

1. The capacity of 32K refers to the number of addressable units, and since K=1024, it means we have 32×1024=32768 addressable units.
2. The size of the data is 16 bits for each addressable unit.
3. The number of address lines required to address 32K units is calculated using the logarithm base 2 of the number of addressable units, which is log2(32768)=15. Therefore, we need **15 address lines**.
4. The number of data lines is equal to the size of the data word, so in this case, we have **16 data lines**.

The 32K*16 bits memory block requires **15 address lines** and **16 data lines**.

- A memory block has 16 address lines and 8 data lines. Specify its capacity.

1. With 16 address lines, the number of addressable units is 2^16=65536
2. Since there are 8 data lines, each addressable unit can store 8 bits (or 1 byte).
3. The total capacity of the memory block is therefore 65536×8
4. Converting this capacity into bytes by dividing by 8 (since there are 8 bits in a byte), we get 65536 bytes.
5. To convert bytes into Kilobytes (KB), we divide by 1024, which yields **64 KB**.

The memory block with 16 address lines and 8 data lines has a total capacity of **64KB**

5. A 20-bit address bus allows access to a memory of capacity

a) 1 MB

b) 32MB

c) 2 MB

d) 64 MB

6. Pipelining improves CPU performance due to

a) reduced memory access time

b) the introduction of parallelism

c) increased clock speed

d) additional functional units

7. In the Intel architecture, there are actually several buses connecting the CPU to the rest of the computer. Which of the following is not such a bus?

a) The control bus

c) The data bus

b) The logic bus

d) The address bus

8. Actual execution of instructions in an embedded system takes place in

a) Control Unit

b) Storage unit

c) ALU

d) None of the above

9. Embedded System functionality include:

a) Stand-alone, small Scaled

b) Stand-alone, Real time, Networked, Mobile

c) Stand-alone, Large Scaled

d) Stand-alone, Small Scaled, Medium Scaled, Large Scale

10. A general purpose microprocessor has on-chip ROM.

a) True

b) False

11. During which phase of the instruction execution cycle is the program counter (IP register) incremented?

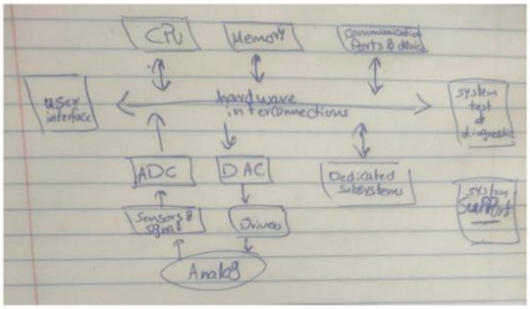
a) decode

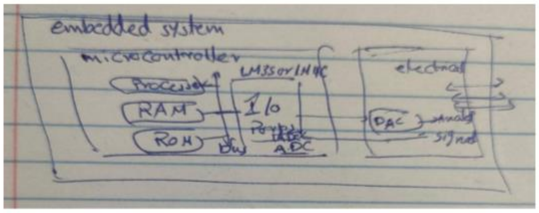
c) operand fetch

b) execute

d) fetch

12.Draw and Explain briefly the hardware elements of an embedded system including the general structure of a microcontroller? List the three types of buses found in microcontroller system and state briefly the purpose of each type of bus. Justify which of them is unidirectional and bidirectional.





1. **Central Processing Unit (CPU)**: Acts as the brain of the microcontroller, executing instructions from the software. It performs arithmetic, logic, control, and data processing.
2. **Memory**: Divided into two main types:
   - **RAM (Random Access Memory)**: Used for temporarily storing data and instructions that the CPU needs while executing programs. It's volatile, meaning it loses its contents when the power is turned off.
   - **ROM (Read-Only Memory)**: Stores the program code and is non-volatile, preserving its contents even without power. This memory is used to store the firmware.
3. **Input/Output (I/O) Ports**: Serve as the interface for the microcontroller to communicate with external devices, sensors, and actuaries. They can be programmed to operate as input or output, allowing the microcontroller to receive data from external sources or send signals to control other devices.
4. **Peripheral Devices**: Include components like timers, counters, ADC (Analog to Digital Converters), and DAC (Digital to Analog Converters), which extend the functionalities of the microcontroller for specific tasks.

Types of Buses in a Microcontroller System

Address Bus: Unidirectional. Carries the memory addresses from the CPU to memory or other devices that the CPU intends to read or write.

Data Bus: Bidirectional. Transports the actual data between the CPU, memory, and other peripherals.

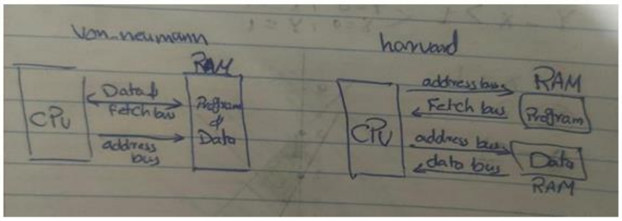
Control Bus: Bidirectional. Carries control signals from the CPU to other components within the microcontroller and vice versa. It includes signals for memory read/write, interrupt acknowledgments, and timing signals.

CSE272 EMBEDDED SYSTEMS

Tut. # 2

13. Van Neumann and Harvard architectures are being employed in the design of embedded processors. Explain briefly the main differences between these two architectures associated with necessary diagrams. Which of these architectures is suitable for high performance CPUs?

- Van Neumann Architecture employs a single bus for both data and instructions, and they are stored in the same memory. This design can lead to a bottleneck, known as the Von Neumann bottleneck, due to the shared use of the bus.
- Harvard Architecture separates the storage and signals for instructions and data, using two distinct memories and buses for each. This separation allows simultaneous access to both instructions and data, enhancing performance.
- Suitable for High-Performance CPUs: Harvard architecture, due to its ability to fetch data and instructions simultaneously, is generally considered more suitable for high-performance CPUs where speed and efficiency are critical.



14. List and briefly discuss the Five design factors affecting the design process of an embedded system related to hardware architecture.

- **Processing Power**: Determining the microprocessor's speed and capability to meet the system's requirements.
- **Memory Requirements**: Amount and type of memory (RAM, ROM) required for the system's operation.
- **Peripherals**: Identifying the necessary peripherals (I/O devices, communication interfaces) and their impact on system design.
- **Reliability**: Ensuring the system is robust and can operate under expected conditions without failure.
- **Future Upgrades**: Considering the ease of making future hardware or software upgrades to extend the system's lifecycle.

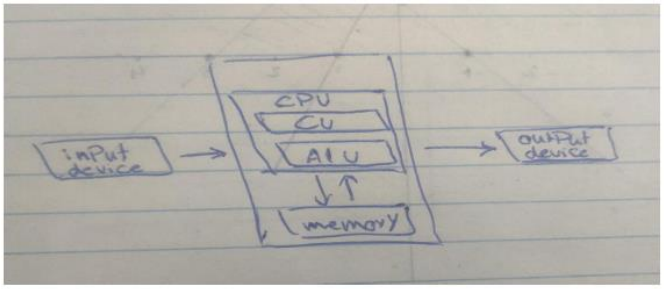
15.Briefly explain the following terms and concepts:

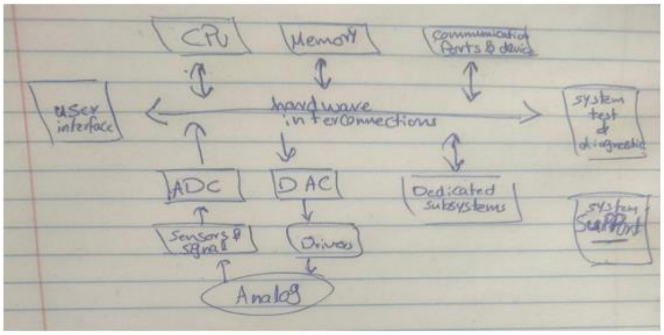
- Half Duplex: Communication method where data transmission can occur in both directions, but not simultaneously.
- Full Duplex: Allows simultaneous data transmission in both directions.
- Firmware: Low-level software that is embedded into the hardware for controlling specific functions.
- IoT and IoT Fields: Internet of Things; a network of interconnected devices able to collect and exchange data. Fields include smart homes, healthcare, agriculture, and manufacturing.
- EPROM: Erasable Programmable Read-Only Memory; a type of memory that can be erased by exposing it to strong ultraviolet light and then reprogrammed.
- CPU Fetch-Execute Cycle: The process where the CPU fetches an instruction from memory, decodes it, and then executes it.
- ALU (Arithmetic Logic Unit): Performs arithmetic and logic operations in a computer.
- CPU (Central Processing Unit): The primary component of a computer that performs most of the processing inside a computer.

16. List four types of the memory addressing modes used in Intel Microprocessor. Explain two of them by an example.

1. Direct Addressing Mode: The memory address of the data is specified directly in the instruction. This mode is straightforward and is particularly useful for accessing global variables or constants that have a fixed memory location.
   - Example: MOV AX, [1234h] moves the data at memory location 1234h directly into the AX register.
2. Register Indirect Addressing Mode: The address of the data is held in a register, and the instruction specifies the register containing the address. This mode allows for flexible data access since the register can be changed dynamically during execution.
   - Example: MOV AX, [BX] uses the address in the BX register to access memory and move the data found there into the AX register.
3. Based Addressing Mode: Combines a base register with an optional displacement (constant value) to calculate the effective address. The base register typically contains a starting address, and the displacement is added to it to access data at a specific offset from this base.
   - Example: MOV AX, [BP+8] accesses data at an address that is 8 bytes beyond the address contained in the BP register and moves it into AX.
4. Indexed Addressing Mode: Uses an index register and an optional scale factor to calculate the effective address, allowing for efficient array access. The index register holds the index within the array, and the scale factor corresponds to the size of each element in the array.
   - Example: MOV AX, [SI*2 + ARRAY] calculates the effective address by multiplying the index in SI by 2 (assuming each element is 2 bytes wide) and adding this offset to the base address of the array stored in ARRAY, then moves the data at the calculated address into AX

17. Draw a diagram of a computer system with external devices. List the major components and explain their functions.





1. CPU (Central Processing Unit): Acts as the brain of the system, executing program instructions and processing data.
2. Memory: Stores instructions and data that the CPU uses during operation. This includes both volatile memory (such as RAM) for temporary storage and non-volatile memory for persistent storage.
3. Communication Ports & Devices: Interfaces for the system to communicate with other systems or devices, which may include serial ports, USB, Ethernet, and wireless communication modules.
4. User Interface: Allows users to interact with the system, which may include components like displays, keyboards, and touch screens.
5. Hardware Interconnections: These are the buses and connections that facilitate communication between the CPU, memory, and other components within the system.
6. ADC (Analog to Digital Converter): Converts analog signals from the external world (such as temperature or pressure readings) into digital values that the CPU can process.
7. DAC (Digital to Analog Converter): Converts digital signals from the CPU into analog signals, which can be used to control devices like motors or to produce audio signals.
8. Sensors & Signal Conditioning: These devices capture data from the environment, such as temperature, light, or motion. Signal conditioning prepares these signals for processing, which may involve amplification, filtering, or converting to a form suitable for the ADC.
9. Actuators & Drivers: These components act upon the external environment based on commands from the CPU. Actuators could include motors, relays, or any device that induces physical action. Drivers are the electronic components that control the power to these actuators.
10. Dedicated Subsystems (ASICs/FPGAs): Application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) are used for specialized processing tasks that are either too resource-intensive for the CPU or require custom logic.
11. System Testing and Diagnostic: These components are used to test and troubleshoot the system during development and maintenance.
12. System Support Subsystems: These may include power supply modules, clock generators, timers, interrupt controllers, and direct memory access (DMA) controllers, all of which support the basic functions of the CPU and system operation.
13. External Analog World: Represents the physical world with which the embedded system interacts, either by sensing parameters or controlling devices.

Marina Reda Abdullah 221101235 AIS

18. True or False. [Justify your answer]

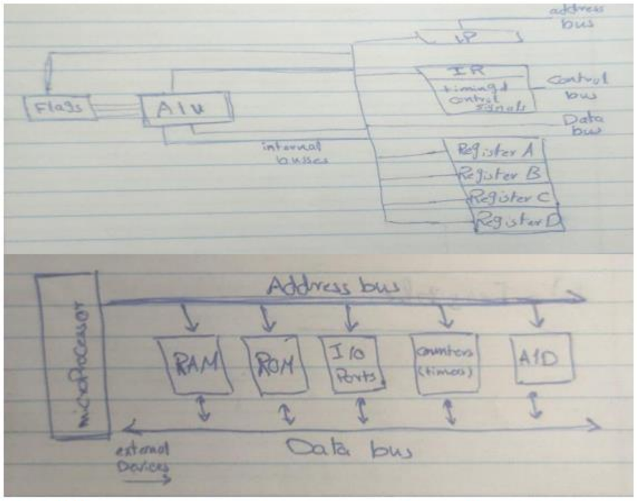
a. Microcontrollers are normally less expensive than microprocessors.

True. Microcontrollers are typically less expensive than microprocessors because they are designed for specific tasks and often have lower processing power.

b. A microcontroller has on-chip I/O ports.

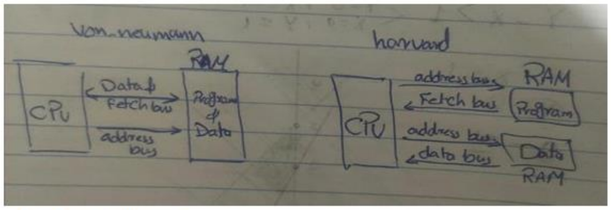
True. A microcontroller typically includes on-chip I/O ports among other features, making it ideal for direct control of other devices.

19. A microprocessor performs a memory read operation, draw and explain the major steps



1. **Fetch Address**: The CPU places the address of the desired memory location onto the address bus. This is the specific location in memory where the data to be read is stored.
2. **Send Read Signal**: The CPU sends a read command on the control bus to signal the memory that it intends to perform a read operation.
3. **Memory Decodes Address**: The memory's control logic decodes the address received via the address bus to locate the exact memory cell.
4. **Data Placed on Data Bus**: The memory retrieves the data from the specified location and places it onto the data bus.
5. **CPU Reads Data**: The CPU reads the data from the data bus into one of its internal registers for processing.

20. Van Neumann and Harvard architectures are being employed in the design of embedded processors. Explain briefly the main differences between these two architectures. Which of these architectures is suitable for high performance CPUs?



- **Van Neumann Architecture**:
  - It features a single storage memory for both data and the program.
  - Requires more memory cycles to extract a full instruction from memory.
  - Special "Table read" instructions are not needed.
  - Pipelining is not possible, making it less efficient compared to the Harvard architecture.
- **Harvard Architecture**:
  - It has separate memories for storage and data.
  - Requires fewer memory cycles to extract a full instruction from memory.
  - Special "Table read" instructions are needed.
  - Pipelining is possible with proper strategies, making it more efficient than the Von Neumann architecture

the Harvard architecture is more efficient due to its ability to support pipelining and because it requires fewer memory cycles for instruction retrieval.

21. The instruction MOV AX, X1[EDI] is an example of

a) indexed addressing

c) direct addressing

b) indirect addressing

d) based addressing

22. What is the result in AL after executing the following instructions? XOR AL, AL OR AL, 80H

a) 80H

c) 00H

b) 88H

d) None of the above

23. Given that the BL register contains 'B', the effect of the following instruction OR BL, 0010 0000b is to

a) clear BL

c) store 'b' in BL

b) store 0010 0000 in BL

d) leave BL unchanged

24.State with a reason, whether the following functional units of a microprocessor belong to control part or execution part:

- **ALU (Arithmetic Logic Unit)**: This is part of the execution unit. It carries out arithmetic and logical operations.
- **Instruction Decoder (ID)**: It is a part of the control unit. It interprets and decodes the instructions to be executed.
- **Data Register (DR)**: Typically associated with the execution unit, as it holds data that may be used or manipulated during instruction execution.
- **Accumulator**: Generally part of the execution unit. It is a register used for arithmetic and logic operations, particularly for storing the results of these operations.
- **Controller/Sequencer**: This is part of the control unit. It manages the control signals that direct the operational flow of the microprocessor, including sequencing the execution of instructions.

25. Instructions are stored in the memory as shown. List and explain the sequence of values in the PC, the address register (AR), data register (DR) if the microprocessor executes the instructions starting with the first one.

Address	Contents	
08H	ADD (1101)	**INSTRUCTION 1**
09H	A (1010)	
0AH	B (1011)	
0BH	SUB (1110)	**INSTRUCTION 2**
0CH	D (1101)	
0DH	9 (1001)	
0EH	SUB (1110)	**INSTRUCTION 3**
0FH	F (1111)	
10H	E (1110)	
…		**INSTRUCTION 4**

Before Instruction 1:

- **PC: 08H (points to the first instruction to be executed)**
- **AR: --**
- **DR: --**

Instruction 1 (ADD A, B):

- **Fetch Instruction:**
  - **PC: 08H (Instruction Address for ADD)**
  - **AR: 08H (Address of the ADD instruction is loaded into AR)**
  - **DR: 1101 (The opcode for ADD is loaded into DR)**
- **Fetch Operand A:**
  - **PC: 09H (Next Address)**
  - **AR: 09H (Address of operand A)**
  - **DR: 1010 (Operand A is loaded into DR)**
- **Fetch Operand B:**
  - **PC: 0AH (Next Address)**
  - **AR: 0AH (Address of operand B)**
  - **DR: 1011 (Operand B is loaded into DR)**

After Instruction 1 is executed, PC will increment to the next instruction.

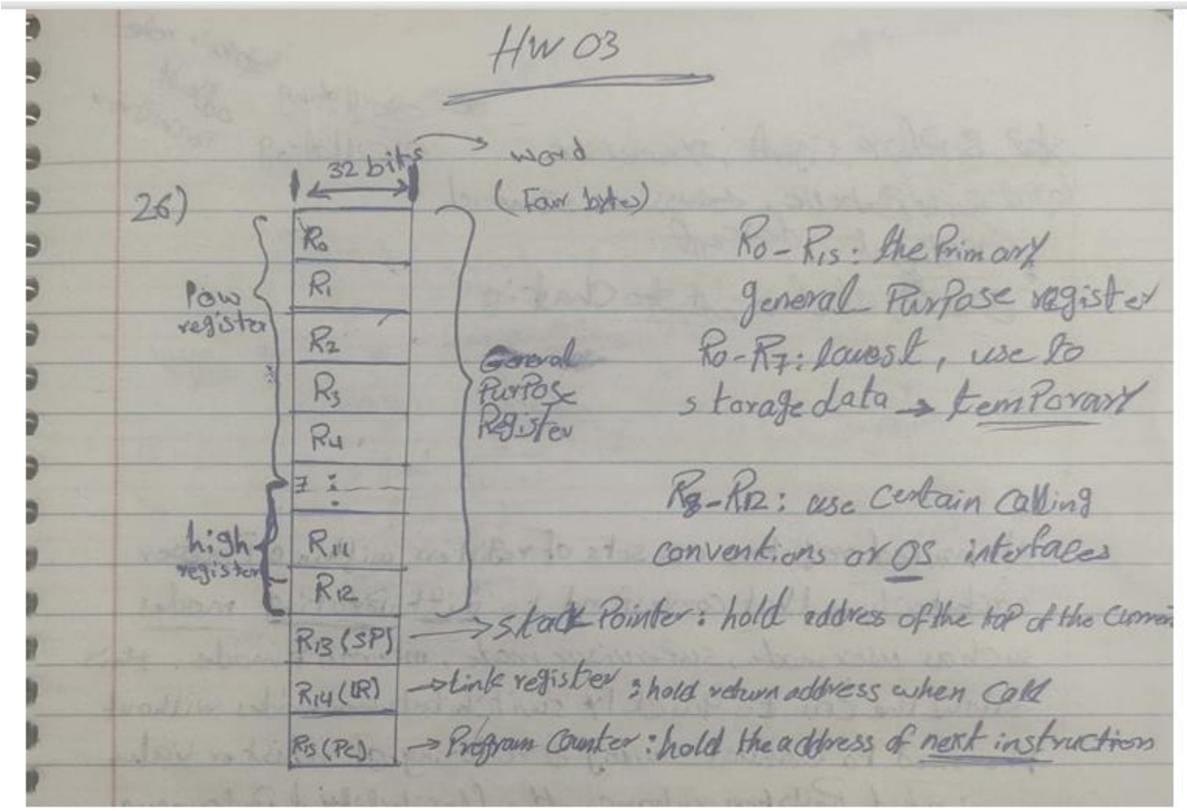
Instruction 2 (SUB D, 9):

- **Fetch Instruction:**
  - **PC: 0BH (Instruction Address for SUB)**
  - **AR: 0BH (Address of the SUB instruction is loaded into AR)**
  - **DR: 1110 (The opcode for SUB is loaded into DR)**
- **Fetch Operand D:**
  - **PC: 0CH (Next Address)**
  - **AR: 0CH (Address of operand D)**
  - **DR: 1101 (Operand D is loaded into DR)**
- **Fetch Operand 9:**
  - **PC: 0DH (Next Address)**
  - **AR: 0DH (Address of operand 9)**
  - **DR: 1001 (Operand 9 is loaded into DR)**

And so on for Instructions 3 and 4:

Instruction 3 (SUB F, E):

- **Fetch Instruction:**
  - **PC: 0EH (Instruction Address for SUB)**
  - **AR: 0EH (Address of the SUB instruction is loaded into AR)**
  - **DR: 1110 (The opcode for SUB is loaded into DR)**
- **Fetch Operand F:**
  - **PC: 0FH (Next Address)**
  - **AR: 0FH (Address of operand F)**
  - **DR: 1111 (Operand F is loaded into DR)**
- **Fetch Operand E:**
  - **PC: 10H (Next Address)**
  - **AR: 10H (Address of operand E)**
  - **DR: 1110 (Operand E is loaded into DR)**


32 bits
word
(Four bytes)
Pow register
high register
General Purpose Register
R_0
R_1
R_2
R_3
R_4
7 ⋮
R_{11}
R_{12}
R_{13} (SP)
R_{14} (LR)
R_{15} (PC)

HW 03

26)

$R_0 - R_{15}$: the Primary general Purpose register

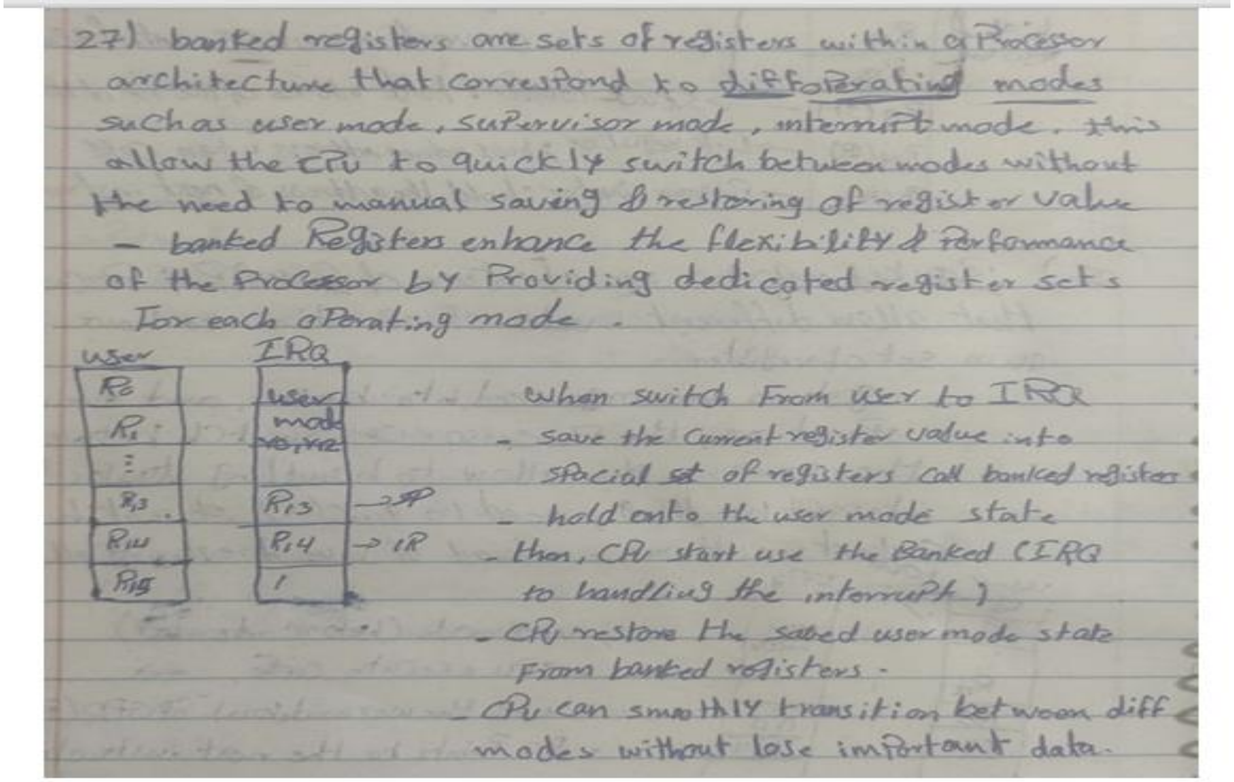
$R_0 - R_7$: lowest, use to storage data → temporary

$R_8 - R_{12}$: use certain calling conventions or OS interfaces

→ stack Pointer: hold address of the top of the Curren

→ Link register: hold return address when Call

→ Program Counter: hold the address of next instructions

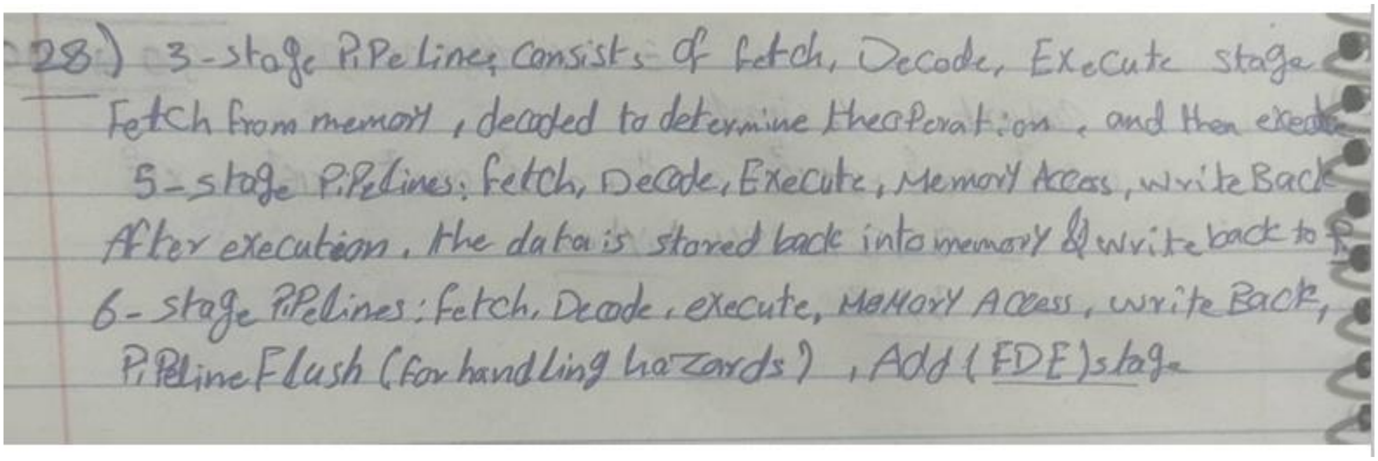

user
R_0
R_1
⋮
R_{13}
R_{14}
R_{15}
IRQ
user mode r0, r12
R_{13}
→ SP
R_{14}
→ LR
/

27) banked registers are sets of registers within a Processor architecture that correspond to diff operating modes such as user mode, supervisor mode, interrupt mode. this allow the CPU to quickly switch between modes without the need to manual saving & restoring of register value

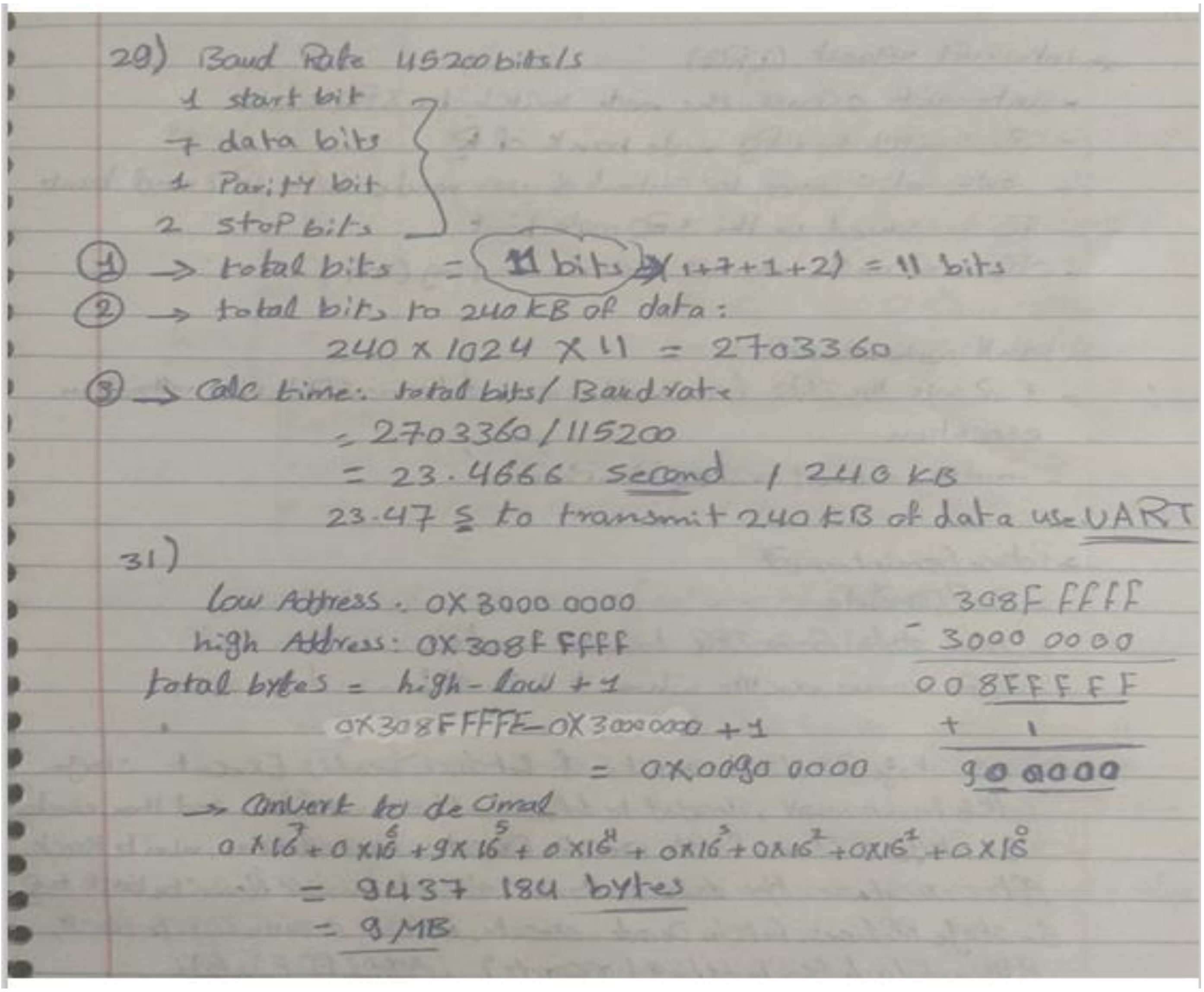
- banked Registers enhance the flexibility & Performance of the Processor by Providing dedicated register sets For each operating mode.

When switch From user to IRQ

- save the Current register value into special set of registers call banked registers
- hold onto the user mode state
- then, CPU start use the Banked (IRQ to handling the interrupt)
- CPU restore the sabed user mode state From banked registers.
- CPU can smoothly transition between diff modes without lose important data.



28) 3-stage PiPeLine; consists of fetch, Decode, Execute stage
Fetch from memory, decoded to determine theoPeration, and then execute
5-stage PiPelines: fetch, Decode, Execute, Memory Access, write Back
After execution, the data is stored back into memory & write back to R
6-stage PiPelines: fetch, Decode, execute, MeMory Access, write Back,
PiPeline Flush (for handling hazards), Add (FDE) stage



29) Baud Rate 115200 bits/s

1 start bit
7 data bits
1 Parity bit
2 stoP bits

① → total bits = 11 bits ⇒ (1+7+1+2) = 11 bits

② → total bits to 240 KB of data:

240 x 1024 x 11 = 2703360

③ → calc time: total bits / Baud rate

= 2703360 / 115200

= 23.4666 second / 240 KB

23.47 s to transmit 240 KB of data use UART

31)

low Address: 0X 3000 0000

high Address: 0X 308F FFFF

total bytes = high - low + 1

0X308FFFFE - 0X3000000 + 1

= 0X0090 0000

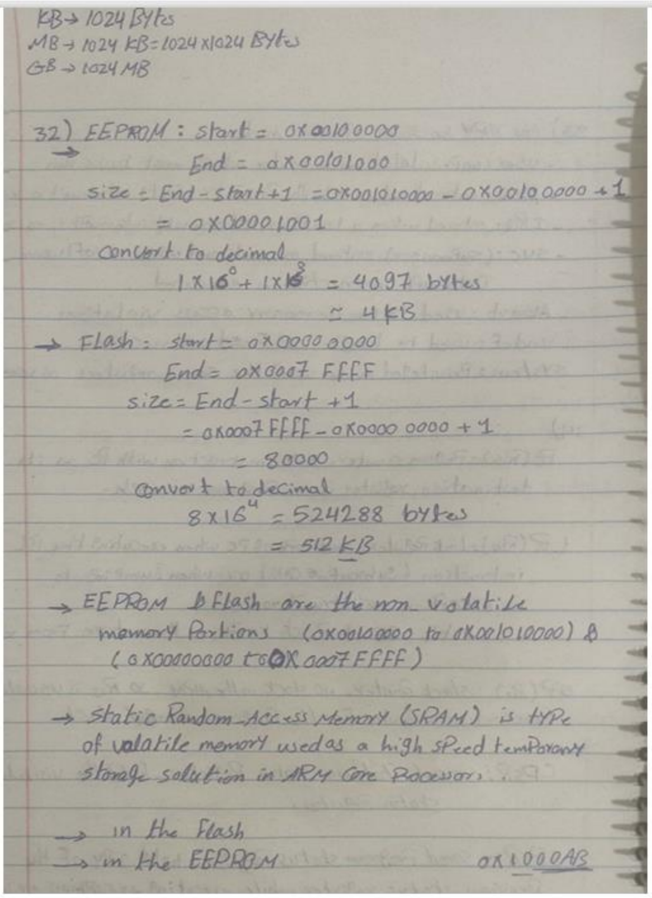
```
  308F FFFF
- 3000 0000
  ---------
  008FFFFF
+        1
  ---------
  900000
```

→ convert to decimal

$0 \times 16^7 + 0 \times 16^6 + 9 \times 16^5 + 0 \times 16^4 + 0 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + 0 \times 16^0$

= 9437 184 bytes

= 9 MB



KB → 1024 Bytes
MB → 1024 KB = 1024 x 1024 Bytes
GB → 1024 MB

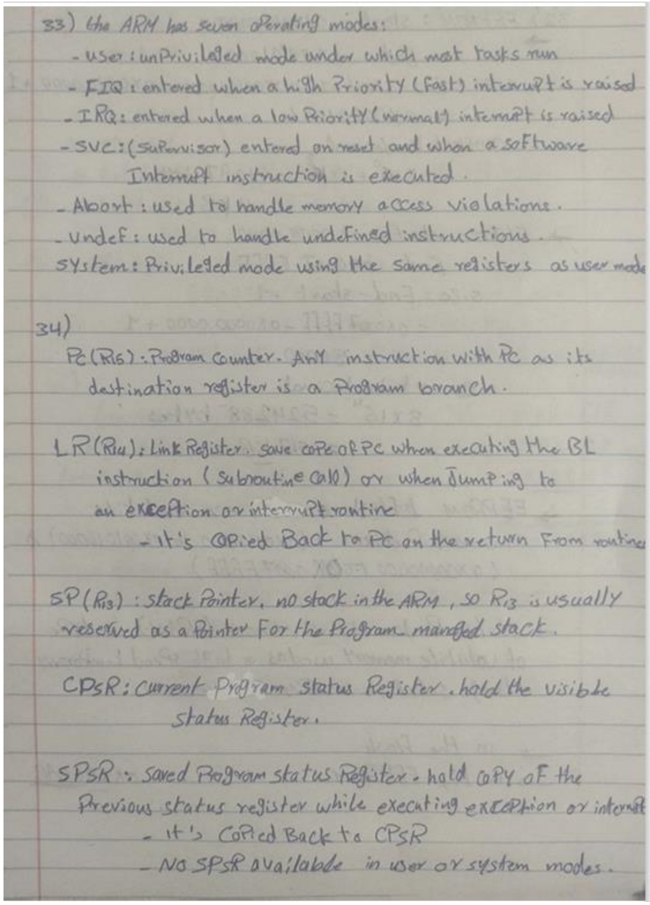
32) EEPROM : start = 0X0010 0000
→ End = 0X00101000
size = End - start + 1 = 0X00101000 - 0X0010 0000 + 1
= 0X00001001
convert to decimal
$1 \times 16^{0} + 1 \times 16^{3}$ = 4097 bytes
≃ 4 KB

→ Flash : start = 0X0000 0000
End = 0X0007 FFFF
size = End - start + 1
= 0X0007 FFFF - 0X0000 0000 + 1
= 80000
convert to decimal
8×16^{4} = 524288 bytes
= 512 KB

→ EEPROM & Flash are the non_volatile memory Portions (0X00100000 to 0X00101000) & (0X00000000 to 0X0007FFFF)

→ static Random Access Memory (SRAM) is type of volatile memory used as a high speed temporary storage solution in ARM Core Processor.

→ in the Flash
→ in the EEPROM 0X1000AB



33) the ARM has seven operating modes:
- user : unPrivileged mode under which most tasks run
- FIQ : entered when a high Priority (fast) interrupt is raised
- IRQ : entered when a low Priority (normal) interrupt is raised
- SVC : (Supervisor) entered on reset and when a software Interrupt instruction is executed.
- Abort : used to handle memory access violations.
- undef : used to handle undefined instructions.
system : Privileged mode using the same registers as user mode

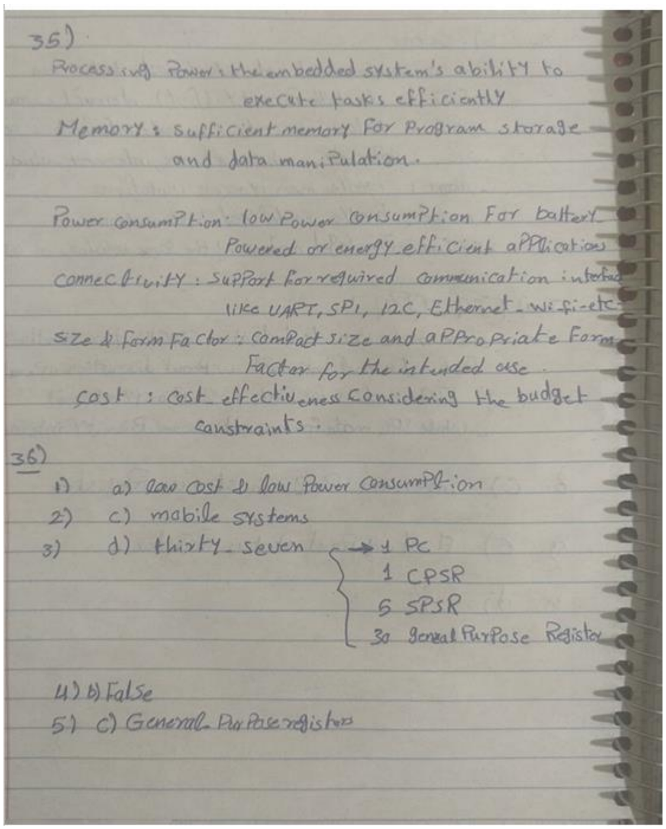
34)
PC (R_{15}) : Program counter. Any instruction with PC as its destination register is a Program branch.

LR (R_{14}) : Link Register. save cope of PC when executing the BL instruction (subroutine call) or when Jumping to an exception or interrupt routine
- it's copied Back to PC on the return from routine

SP (R_{13}) : stack Pointer. no stack in the ARM, so R_{13} is usually reserved as a Pointer for the Program manged stack.

CPSR : Current Program status Register. hold the visible status Register.

SPSR : saved Program status Register. hold copy of the Previous status register while executing exception or interrupt
- it's Copied Back to CPSR
- No SPSR available in user or system modes.

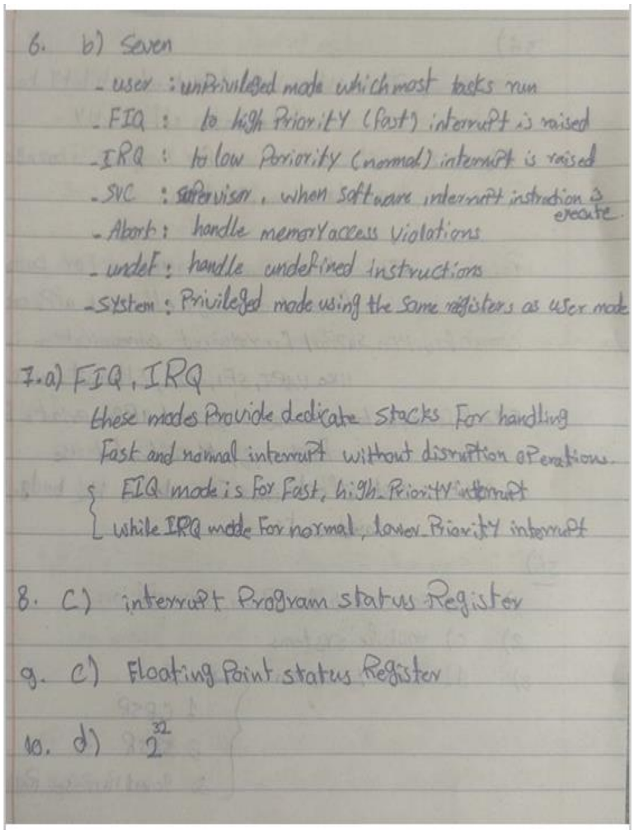


35)
Processing Power : the embedded system's ability to execute tasks efficiently
Memory : sufficient memory for Program storage and data manipulation.

Power consumption : low Power consumption for battery Powered or energy efficient applications
connectivity : support for required communication interface like UART, SPI, I2C, Ethernet, wi-fi-etc
size & form Factor : compact size and appropriate Form Factor for the intended use.
cost : cost effectiveness considering the budget constraints.

36)
1) a) low cost & low Power consumption
2) c) mobile systems
3) d) thirty_seven → 1 PC
1 CPSR
5 SPSR
30 general PurPose Register

4) b) False
5) c) General PurPose registers



6. b) Seven
- user : unPrivileged mode which most tasks run
- FIQ : to high Priority (fast) interrupt is raised
- IRQ : to low Poriority (normal) interrupt is raised
- SVC : supervisor, when software interrupt instruction is execute.
- Abort : handle memory access violations
- undef : handle undefined instructions
- system : Privileged mode using the same registers as user mode

7. a) FIQ, IRQ
these modes Provide dedicate stacks for handling Fast and normal interrupt without disruption operations.
FIQ mode is for fast, high_Priority interrupt
while IRQ mode for normal, lower Priority interrupt

8. C) interrupt Program status Register

9. c) Floating Point status Register

10. d) 2^{32}