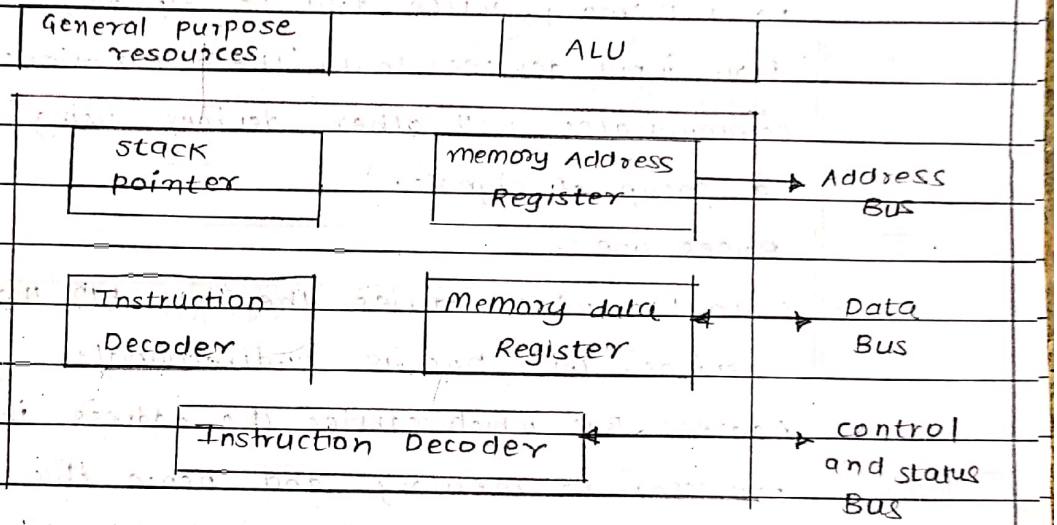
Set B

Q.1. Explain the internal architecture of a general purpose processor and on basis of the instruction between memory and CPU, classify and draw the 3 promient architectures.

→ ① The GPPs are further classified as micro-controller and micro-processors. A microprocontroller has memory and other peripherals on the chip itself, Hence it is best choice for small embedded systems. On the other hand, a microprocessor is more powerful beit requires a large number of external components.

② Internal architecture of a processor.



The CPU consists of :

① Arithmetic Logic Unit (ALU) which performs arithmetic and logic operations.

② General purpose registers. This registers constitute the processor's internal memory. The number of registers varies from processor to processor. Registers contain the current data and operands that are being manipulated by the processor.

③ Control unit that fetches the instructions from memory, decodes them and executes them. Control unit consists of

- Instruction pointer that points to the next instruction to be executed. Instruction pointer also called program counter.
- Stack pointer that points to the stack in memory. In external

contents of registers are transferred to this stack. The processor keeps track of the next free location in the stack through stack pointer.

- Instruction decoder that decodes the instructions.
- Memory Address Register and Memory Data Register.

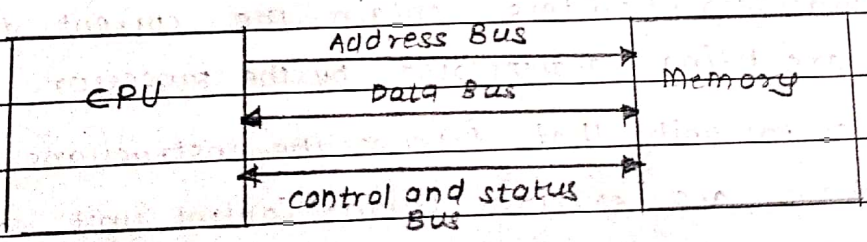
- Processor Architectures :-

In addition to a manipulating data, a processors job is to read data and instructions from memory and read and write data to a memory, write data to o/p devices and read data from input devices. To do these functions, the processor communicates with other devices using three buses, a bus being a group of signals.

Buses are :-

- Data Bus which carries the data b/th the processor and other devices. This bus is bi-directional.
- Address Bus which carries the address information from the processor to memory and hence this bus is unidirectional.
- Control and status bus which carries control/status information such as whether the operation is read or write, Indication of address error, as well as processor reset signal clock input and interrupt signal

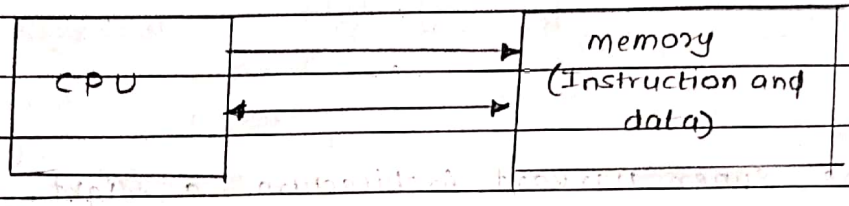
A] Interaction between CPU and memory



Based on the number of memory and data buses used, there are three types of architectures for the processors.

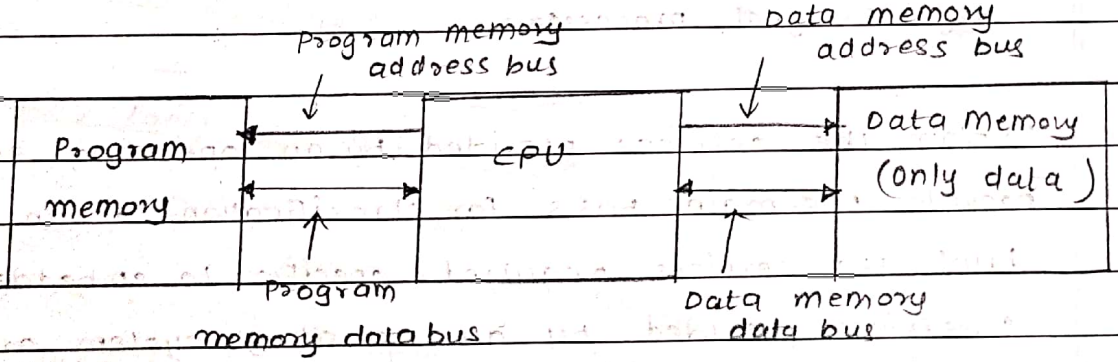
These are :- ① Von Neuman Architecture
② Harvard Architecture
③ Super Harvard Architecture

① Von Neumann Architecture :-



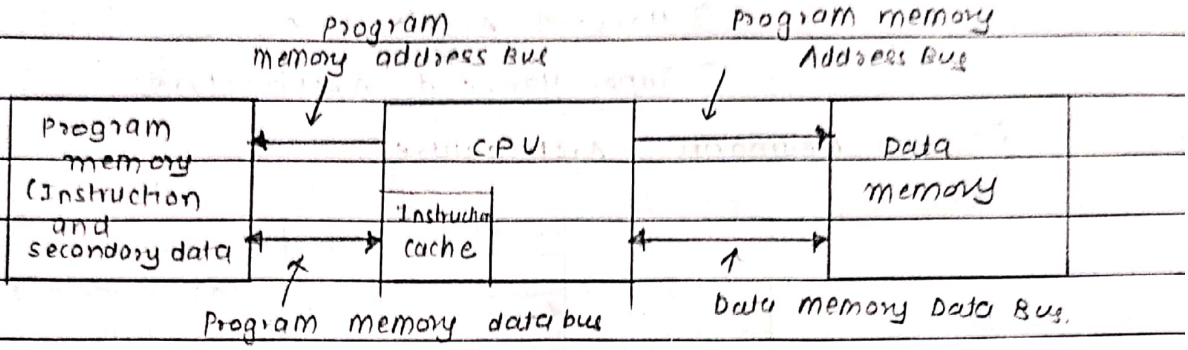
The Von-neumann architecture is the most widely used architecture, this architecture has one memory chip which stores both instructions and data. The processor interacts with the memory through address and data buses to fetch instructions and data.

② Harvard Architecture :-



In this architecture there are two seperate memory blocks. One is program memory and other is data memory. Program memory stores only instructions and data memory stores only data. Two pairs of data buses are between the CPU and the memory blocks. Program memory blocks address bus and program memory data bus are used to access the program memory. Data memory address bus and data memory data bus are used to access the data memory.

(5) Super - Harvard Architecture:



The super Harvard Architecture a slight but significant modification of the Harvard architecture. In Hardware architecture the data memory is accessed more frequently than the program memory. Therefore, SHARC provision has been made to store some secondary data in the program memory to balance the load on both memory blocks. This architecture, proposed by analog devices, is used extensively in digital signal processors.

Q.1. Explain the services provided by an operating system and provide the main basis for classification of an OS provide in brief the services required specific to embedded OS.

→ • Services provided by an operating system are :- (05)

1) Process / Task Management
2) memory management
3) I/O management + File system management
4) Providing services to applications.
5) Providing user interface to access OS via API

Operating System

single Task OS vs multitask OS	single user OS vs multi-user OS	Commands vs GUI
MS DOS vs Windows & Unix	MSDOS & Windos vs Unix	MSDOS vs windows & unix

- Services Required for embedded operating system :-

1) Reliability (99.999 % Downtime per year)
2) Multi-tasking with time constraints
3) Small footprint - (No frills memory hence no GUI)
4) Support Diskless systems - (No secondary storage.)
5) Portability - (Varies of processor unlikes Intel)
6) Scalability - (8 bit to 64 bit)
7) Support for API (function calls) via POSIX.

1) Reliability :- The OS is an embedded system has to be very reliable. If the system has to have an availability of 99.999 % of the time, the downtime per year is just about 5 minutes.

2) Multi Tasking with time constraints : Embedded systems need to support multi-tasking. All operating systems used in embedded systems support this feature. But then, the task management has to be done efficiently to meet the real-time performance requirements.

3) Small footprint :- As the memory devices have limited capacity in embedded systems, they have little memory for the OS. Many frills provided by operating systems such as Windows are not provided in embedded operating systems. The memory occupied by the operating system is known as the 'footprint'. The footprint should be very small for an embedded OS.

4) Support diskless systems :- Unlike the desktop computers, embedded systems may not have have secondary storage such as hard disk or CDROM. The embedded OS along

with application software will reside on a memory chip. File System management is not mandatory in embedded systems.

5) Portability :- A variety of processors are available for developing embedded systems unlike desktop computers whose processors are predominantly the Intel family processors. So, an important requirement of embedded operating systems is portability.

6) Scalability :- The embedded operating systems may be used on an 8-bit micro-controller or a powerful 64-bit microprocessor. So, scalability is very important for embedded operating systems.

7) Support for standard API :- Application software is developed using the Application Programming Interface (API) of the operating system. API is a set of function calls. An application developed for one OS may not be portable to another OS. To achieve portability, IEEE standardize the API called Portable operating System Interface (POSIX).

Q.2. Explain the design challenges, design trade-offs and design metrics in the Hardware-software co-design Principles of an Embedded System using a suitable venn diagram where needed.

→ * Design metrics :- While desiging an embedded system following points are to be considered

i) NRE (Non-Recurring Expentidure) cost :-
This includes the costs that are occurred only once and are not for each system. This includes the development cost of the system are other investments that are to be done.

ii) Unit Cost :- This is the cost of each unit. The cost includes the components costs and the manufacturing cost involved in it. It excludes the cost of development of the system for the 1st time.

iii) size :- size of the system is measured in terms of
i) Physical space required
ii) memory capacity often measured in bytes for software and data.
iii) No. of logic gates in the hardware.

iv) Performance :- The performance of system is measured by the time required to execute a task. Smaller the execution time means higher performance.

v) Power :- The power consumption is also an imp attribute for the handled ES or ES used in field i.e. powered by the battery. If the power consumption is less the battery lasts longer and hence it is considered as a better system.

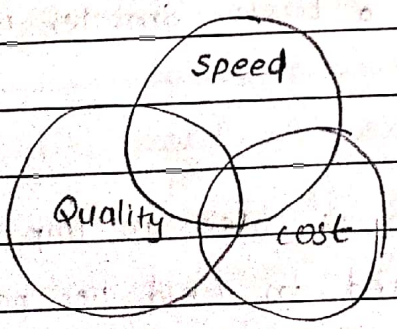
vi) Flexibility :- A system that can have changes made in it without investing in research or the NRE cost are said to have flexibilities. Hence a system that has more flexibilities is better.

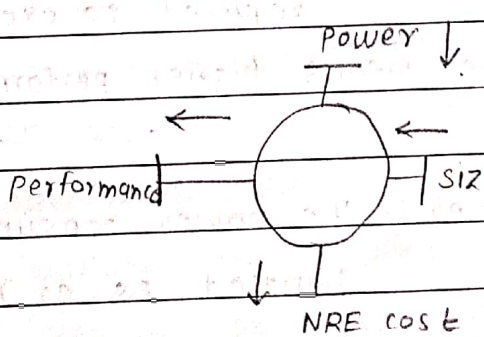
vii) Time-to-prototype :- The time required to make a proto-type of the system is very imp. The time required should not be very high, else another competitor may bring the product in the market.

viii) Time-to-market :- Time-to-market depends on design time, manufacturing time and testing time.

ix) Maintability :- Modification in the system can be done by adding, or updating software, data and hardware.

x) System and user safety :- This metric ensures that the system will not cause any harm. System safety means safety interns of accidental fall from hand or table, theft eg. a mobile phone locking ability and tracking ability. User safety means safety for user when using a product.





Design challenges

Design trade-offs

- Design Trade-Offs :-

① NRE cost :- It is the non-recurring engineering cost. It is only one time cost including research, prototype preparing, PCB design cost.

② Size :- Size of the system is measured in the term of physical space required, minimum capacity measured for (S/W) bytes, No. of logic gates used in H/W.

③ Power :- ES are powered by batteries. If power consumption is less the battery lasts longer and hence it is considered as a better system.

④ Performance :- It is measured by time required to execute a task. Smaller the execution time, means higher performances.

These metrics typically compete with eachother. Improving one often leads to degrading another. For eg. if we reduce the size of ES, it will affect the performance of system. If we reduce the power consumption of the system, the NRE cost can be affected. This phenomenon is compared to a wheel with these 4 pins called as design tradeoffs.

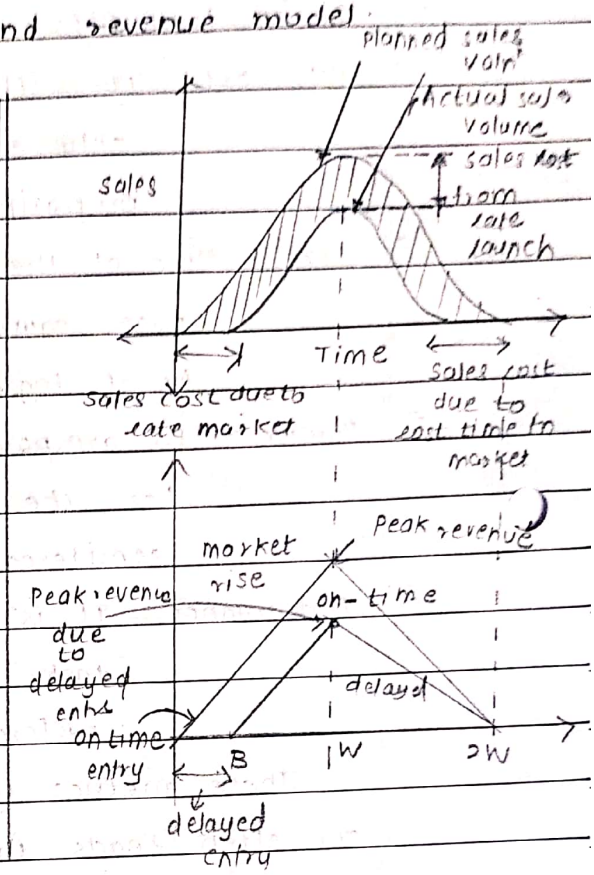
- Design challenges :- Any ES when being designed, 3 factors are considered to be important including Quality of the system, speed of the system and it's cost.

For any system, its quality of the materials used and it's speed of execution should be as high as possible. If there were to be implemented, the overall cost of the system will increase which has to be avoided.

So the designer has to design a system very effectively where its quality and speed cannot be compromised for reasonable cost. All the 3 factors will be perfectly balanced.

Q.2. Product life cycle or sales and revenue model.

→ In this model, the life cycle of a product right from its expected release to its end is described. The graphs describe and composed between the sales and revenue if the product was launched on time scheduled time & if it was released on delayed time. The loss of sales and revenue loss is also found out.



product life cycle = 2W

max sale of = W

Area of triangle = total revenue.

Area (on time) = $\frac{1}{2} \times 2W \times W = W^2$

Area (delayed) = $\frac{1}{2} \times (2W-D) \times (W-D)$ = B

Loss = $W^2 - \frac{(2W-D)(W-D)}{2} = \frac{D(3W-D)}{2}$

% revenue loss = $\frac{\text{delayed}}{\text{original}} \times 100 = \frac{D(3W-D)}{2W^2} \times 100$

choice A):- X = 25,000, Y = 5,000,

Total cost = X + NY = 5,00,25,000

Per unit cost = (X/N) + Y = 5002.5

choice B):- X = 45,000, Y = 3500

Total cost = X + NY = 3,50, 45,000

Per unit cost = (X/N) + Y = 3504.5

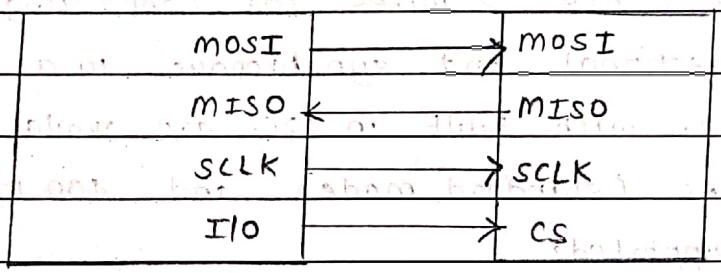
Choice C):- X = 36,000 Y = 4,200

Total cost = X + NY = 4,20, 36,000

Per unit cost = (X/N) + Y = 4203.6

1. Explain SPI and I2C interface to the processor.

→ SPI has developed by motorola, peripheral devices such as memory chips, potentiometer, ADCs and DACs, Real-time-clock etc. are provided with SPI interface so that they can be interfaced to the processor. The processor generates the clock and the peripheral use the clock to synchronize its acquisition of the data.



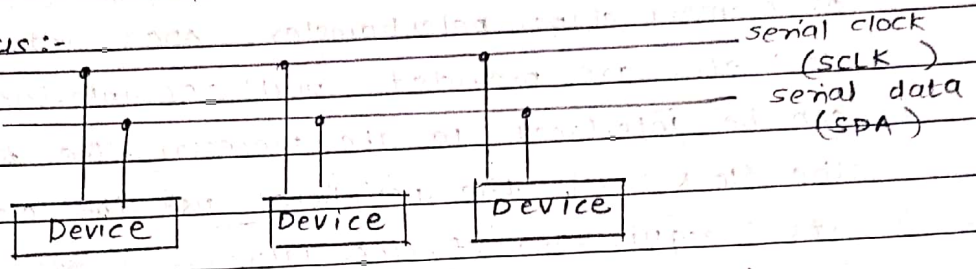
SPI use four type of signals for interfacing peripherals to the processor

- Master out slave in (MOSI)
- master in slave out (MISO)
- serial clock (SCLK)
- chip select (CS) for the peripherals.

SPI is based on the master-slave protocol. The processor acts as the master and the peripheral acts as the slave the processor and the peripherals are connected using these 4 signals. Both the master and slave contain shift registers. The master sends a byte to the slave on MOSI line, and slave sends its register contents on MISO line. Both read and write can be simultaneously. If the master has to read a byte from the slave, it has to write a dummy byte to initialize the slave for transmission.

This is a synchronous protocol for communication between the processor and peripheral

I2C Bus :-



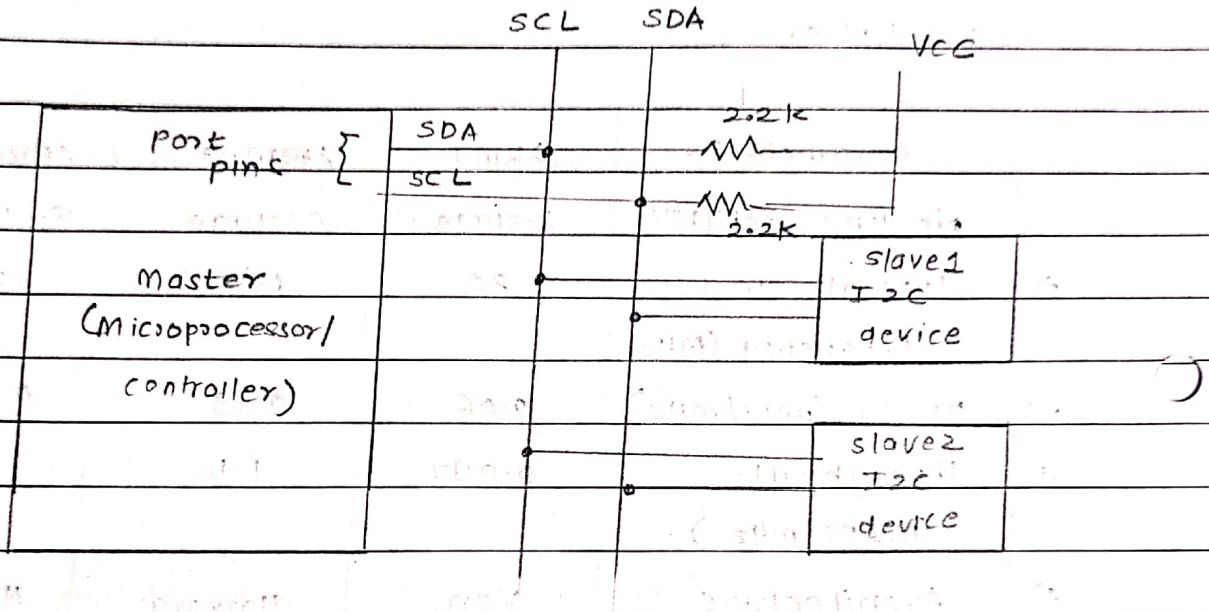
I2C Bus uses 2 wires for connecting devices. The bus is bidirectional and synchronous to a common clock. Micro-controller with built in I2C are available. Data rates of 100 kbps (standard mode) and 400 kbps (fast mode) are supported.

The Bus consists of two lines - serial clock (SCL) and serial data (SDA). Both lines remain high when not in use. A device using the bus drives the line low. Each device has a unique address of 7-bits or 10-bits. If 7-bits are used, 128 devices can be connected to the bus. A device can acts as a master or slave. Transmitting device is the master and the reducing device is the slave. The same line is used for master transmission and slave response. I2C bus is a multi-master bus, more than the one device can acts as a master. We can interface a RTC such as philips PCF8353 a display of matrix.

I2C is a shared Bus system to which many no. of I2C device can be connected. Devices connected to the I2C bus can act as either master device or 'slave' device

Master and slave devices can acts as either transmitter or receiver. Regardless whether a master is acting as transmitter or receiver, the synchroniz...

clock signal is generated by the 'master' device only. I2C supports multi masters on the same bus. The following diagram illustrate the connection.



The I2C bus interface is built around an input buffer and an open drain or collector transistor. When the bus is in the idle state, the open drain/ collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the High impedance state.

Q. Compare ARM 7, 9, 10 & 11 or compare ARM m0+/- m3, m4 on the basis of following:

- Pipeline depth, clock frequency, power, throughput, architecture, multiplier.

	Parameters	ARM7	ARM 9	ARM 10	ARM 11
1)	pipeline depth	3-stage	5-stage	6-stage	8-stages
2)	Typical clock frequency (MHz)	80	150	260	335
3)	power (mW/mHz)	0.06	0.19	0.50	0.040
4)	Throughput (MIPS/MHz)	0.097	1.1	1.3	1.2
5)	Architecture	Von Neuman	Harvard	Harvard	Harvad
6)	Multiplier	8*32	8*32	16*32	16*32

- Compare ARM m0+/- , m3, m4

	Parameters	m0+/-	m3	m4	m0
1	pipeline depth	2-stage	3-stage	3-stage	3stage
2	Power (watt)	474	141	151	66
3	Throughput (MIPS/MHz)	1.37	1.89	1.95	1.21
4	Architecture	ARMV6-M Ven-neumman	ARM V7-M standard	ARM-7M Harvard	ARMV6-M Von-neuman

Q.4. Write a Branch Instruction and conditional execution instruction to execute the following condition & compare the code density.

- IF contents of register r5 are equal to 20 add the contents of r5 to register r2 after logically shifting r2 by 4 bits & store this results in r0.

Explain the difference in the below instructions

- LDR r0, [r1, #4]
- STR r0, [r1, #12]
- LDR r0, [r1, #4]!
- STR r0, [r1, #12]!
- LDR r0, [r1], #4
- STR r0, [r1], #12
- LDRSH r0, [r1]
- STRB r0, r1

→ A) Using Branch Instruction :-

```
    CMP  r5, #20
    BEQ  SKIP        ——— (IF zero flag is set)
    ...
    ...
SKIP : ADD r0, r5, r2, LSL#4
```

B) Using conditional execution instructions :-

```
    CMP   r5, #20
    ADDEQ r0, r5, r2, LSL#4
```

By comparing code density of case (A) and (B) we can see that by using conditional instructions it requires less code/ instructions which makes code more dense.

Compare to Branch instruction extra machine cycle gets wasted.

① LDR r0, [r1, #4]

→ r0 ← [r1+4] ; r0 = mem[r1+4]

r0 gets data from location pointed by [r1+4].

② STR r0, [r1, #12]

→ [r1+12] ← r0 mem[r1+12] = r0

Store word to memory location pointed by r1 then update r1 to r1 offset

③ LDR r0, [r1, #4]!

→ r0 = mem[r1+4] , r1 = r1+4

Here r1 is gets auto-incremented and r0 gets data from location pointed by [r1+4].

④ STR r0, [r1, #12]!

→ [r1+12] ← r0 , r1 = r1+4

Store word to memory with Autoincrementing r1.

⑤ LDR r0, [r1], #4

→ r0 = mem[r1], r1 = r1+4

Read word from memory r1 to r0, then update r1 to r1 offeset (post indexing).

⑥ STR r0, [r1], #12

→ mem[r1] = r0, r1 = r1+12

⑦ LDRSH r0, [r1]

→ Read and signed extended half word from memory r1 to r0.

r0 = mem16[r1]

⑧ STRB r0, [r1]

→ store byte to memory r1

mem8[r1] = r0

Q.4. Consider a 8 stage pipeline with stage delays 15, 22, 26, 14, 19, 25, 12, 18 ns. It is required to process 8000 data items in the pipeline. What is the minimum time required to process the data items? If the latch delay is to be considered as 6ns for each stage what is the maximum clock frequency with which the pipeline can operate.

→ Max. delay of stage = τ_m = 26ms

τ (clock period of pipeline) = $\tau_m + \tau_l$ (latch delay)

= 26ns + 6ns

τ = 32ns

N (data items) = 8000

K = No. of stages of pipeline = 8

Time require to process data = $[(K-1)+N]\tau$

= [7 + 8000] × 32 ns

= 2.56×10^{-4}

= 256.22 ms

Maximum clock frequency = $\frac{1}{\tau} = \frac{1}{32ms}$

= 31.25 MHz

B] If the memory location 1000, 1004, 1008 contains the data 15, 18, 24 and r1 is initialized to 1000 The contents of register r2 after execution of following code segment.

→ A) LDR r5, [r1]

r5 ← #15 ; r5 is loaded with 15 which is present in memory location pointed by [r1].

r5 = mem [r1]

r5 = 15

→ LDR r6, [r1, #4]

r6 ← [r1+4] ; r6 = mem[r1+4]

= mem[1004]

r6 = 18

→ LDR r7, [r1, #8]

r7 = mem[r1+8]

= mem[1008]

r7 = 24

- ADD r2, r5, r6, LSL #3

⇒ r2 = r5 + (r6 << 3)

r6 = 0000 0000 0001 0010

LSL #3 ⇒

r6 = 0000 0000 1001 0000

= 144

r2 = 15+144 = 159

- SUB r2, r2, r7, LSR #2

r7 → 0000 0000 0001 1000

LSR #12

r7 → 0000 0000 0000 0110

r7 = 6

r2 = r2 - (r7 >> 2)

= r2 - 6

= 159 - 6

r2 = 153