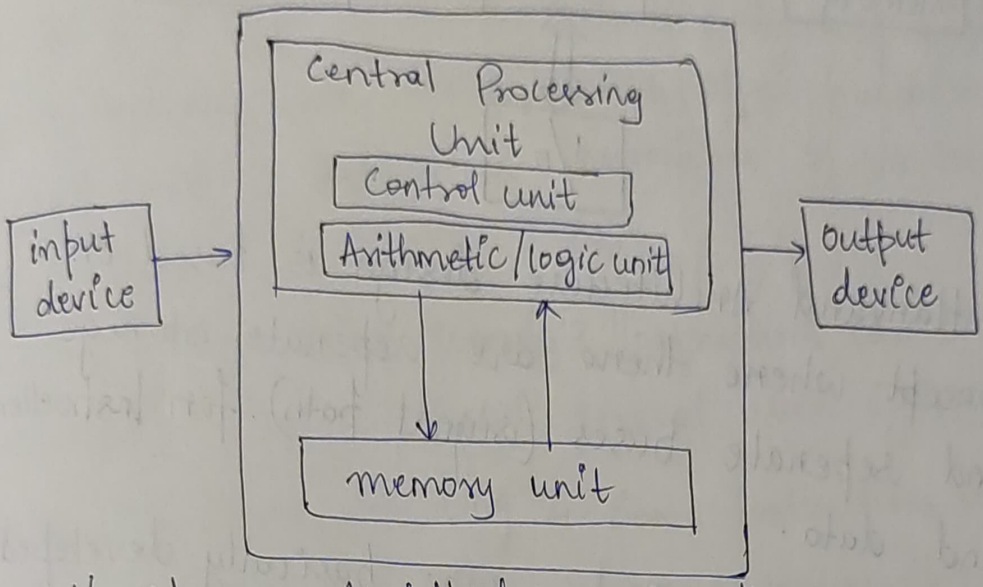
Assignment-01

04/12/21

① With a neat diagram, explain in brief VonNeuman and Harvard Architecture.

⇒ **VonNeuman Architecture :-**



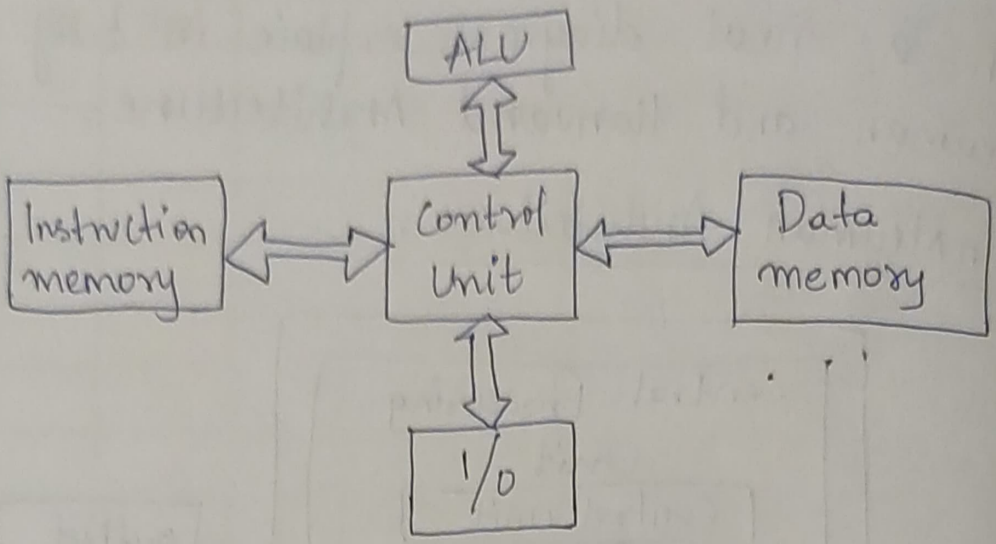
→ Von Neuman Architecture also known as the "Von Neuman model" or "Princeton Architecture".

→ The design is based on the Concept of stored program Computers where program data and instruction data are stored in the same memory.

→ VonNeuman Architecture Comprises 3 Components:

- A processing unit that Contains an Arithmetic logical unit (ALU) & Processor registers.
- A Control unit that Contains an instruction register and pgm Counter.
- Memory that stores data and instructions.
- Input and Output mechanisms.

⇒ Harvard Architecture :-



→ Harvard Architecture design is based on concept where there are seperate storage and seperate buses (signal path) for Instruction and data.

→ Harvard Architecture was basically developed to overcome the Von Neuman bottleneck. ie., in Von Neuman Architecture, instruction fetch & data operation cannot occur at the same time because they share a common bus, which was often limiting the performance of the system.

→ An instruction is executed in single clock cycle.

→ CPU can access instructions and read/write at the same time.

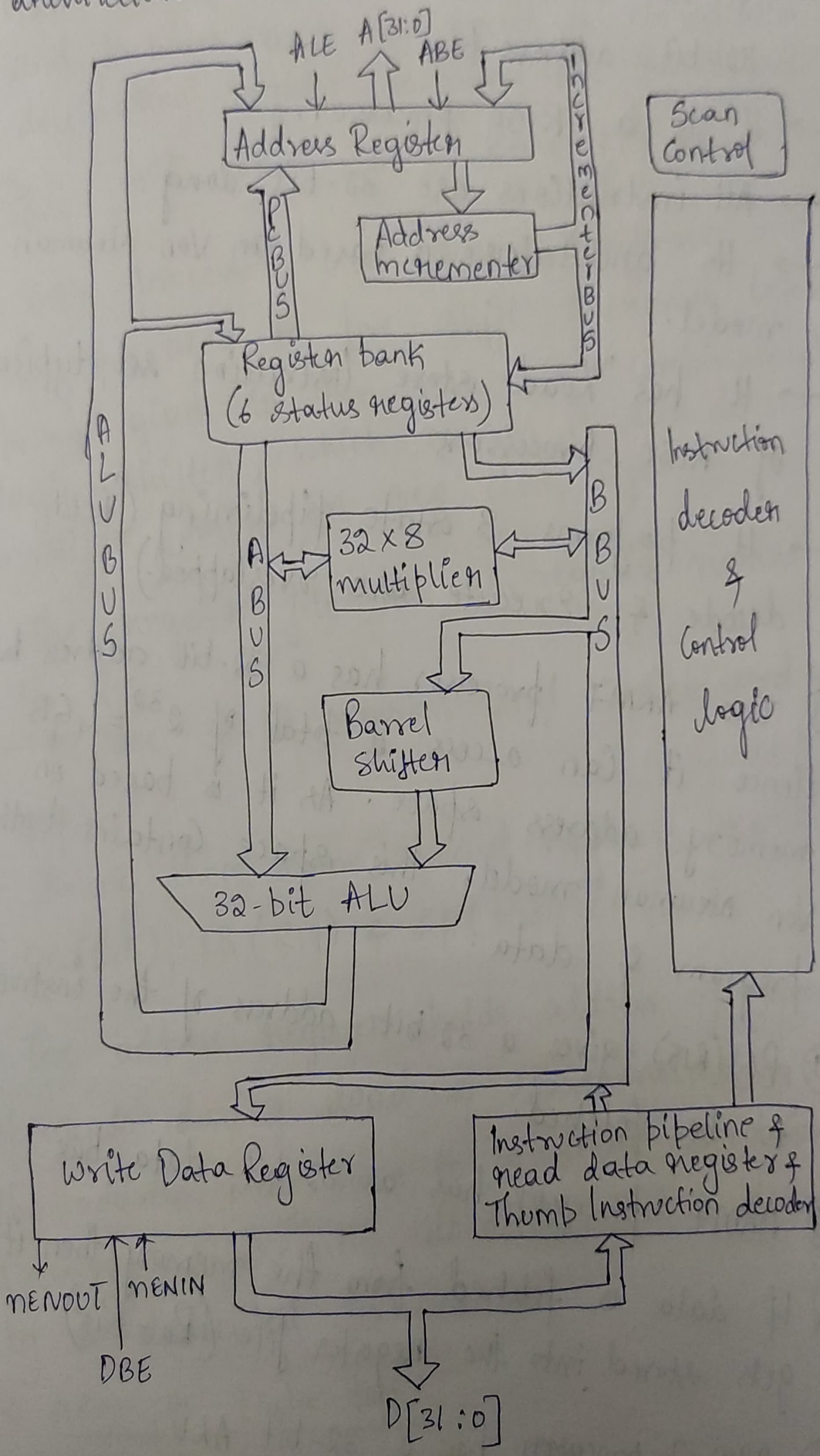
(2) List the differences between RISC and CISC.

⇒

RISC	CISC
1. It stands for Reduced Instruction Set Computer.	1. It stands for Complex Instruction Set Computer.
2. It is a type of μprocessor architecture that uses a small set of Instructions of Uniform length.	2. It is a type of processor which offers hundreds of Instructions of Variable size.
3. Simple Instructions, generally executed in one clock cycle.	3. Instructions can take several clock cycles.
4. No Instruction with long execution time due to very simple Instruction set.	4. Some Instructions with long execution time.
5. Fixed-length encodings of the Instruction are used.	5. Variable-length encoding of the Instructions.
6. Simple addressing formats are supported - only base & displacement addressing is allowed.	6. Multiple formats are supported for specifying operands. Supports different combinations of displacement, base & index registers.
7. RISC does not support array. Arithmetic & logical operations only use register operands.	7. CISC supports array.

8. Registers are being used for procedure arguments and return addresses.	8. The stack is being used for procedure arguments & return addresses
9. Software-Centric design - High level Compilers take on most of the burden of coding many software steps from the programmer.	9. Hardware-Centric design - The Instruction set architecture does as much as possible using hardware Circuitary.
10. Heavy usage of RAM.	10. More efficient use of RAM than RISC.
11. Hardwired Control	11. Microcode Control.
12. Pipelining takesplace	12. No pipelining takesplace.

(3) With a neat diagram, explain ARM7 architecture.



ARM7TDMI BLOCK DIAGRAM

→ ARM7 is a 32-bit micro Controller.
→ It has 32-bit data bus and 32-bit ALU & 32-bit address bus.
→ It is a RISC processor.
→ All instructions are 32-bit long.
→ Its architecture is based on Von-Neuman model.
→ It has "Load-store" Instruction set, typical of RISC processors.
→ It performs 3 state pipelining (Fetch, decode & Execute are overlapped.)

① the ARM7 processor has a 32-bit address bus. Hence it can access a total of $2^{32} = 4GB$ memory address space. As it is based on Von Neuman model, this space contains both program & data.

② PC (R15) gives a 32-bit address of the instruction to be fetched.

③ ARM7 processor has a 32-bit data bus.

④ If data is fetched from the memory then it gets stored into the register file ($R_0 - R_{15}$).

⑤ ARM-7 processor has a 32-bit ALU.

⑥ ALU does triadic operations - Two operands from register R_M & R_N are obtained using A and B buses and the result is stored in a destination register R_D using the result bus.

Eg:- ADD R_0, R_1, R_2; $R_0 \leftarrow R_1 + R_2$

⑦ ARM processor has barrel shifter: This is used to pre-shift operands before being given to the ALU. It is a combinational logic shifter, which means it can do multiple shifts in one clock cycle.

⑧ the register file has 16-registers named R_0-R_{15}.
- All are 32-bit registers and can be used as GPR's.
- Among them some special registers like PC(R_{15}), LR(R_{14}) & SP(R_{13}).

⑨ The address register holds address of memory operands during load & store instructions. The address may have to be incremented for accessing subsequent locations. This is done by a dedicated Incrementer.

⇒ ARM 7 Data types:

→ Byte : 8-bits
→ Halfword : 16-bits
→ word : 32-bits

(4). With a neat diagram, explain the programming model of ARM.

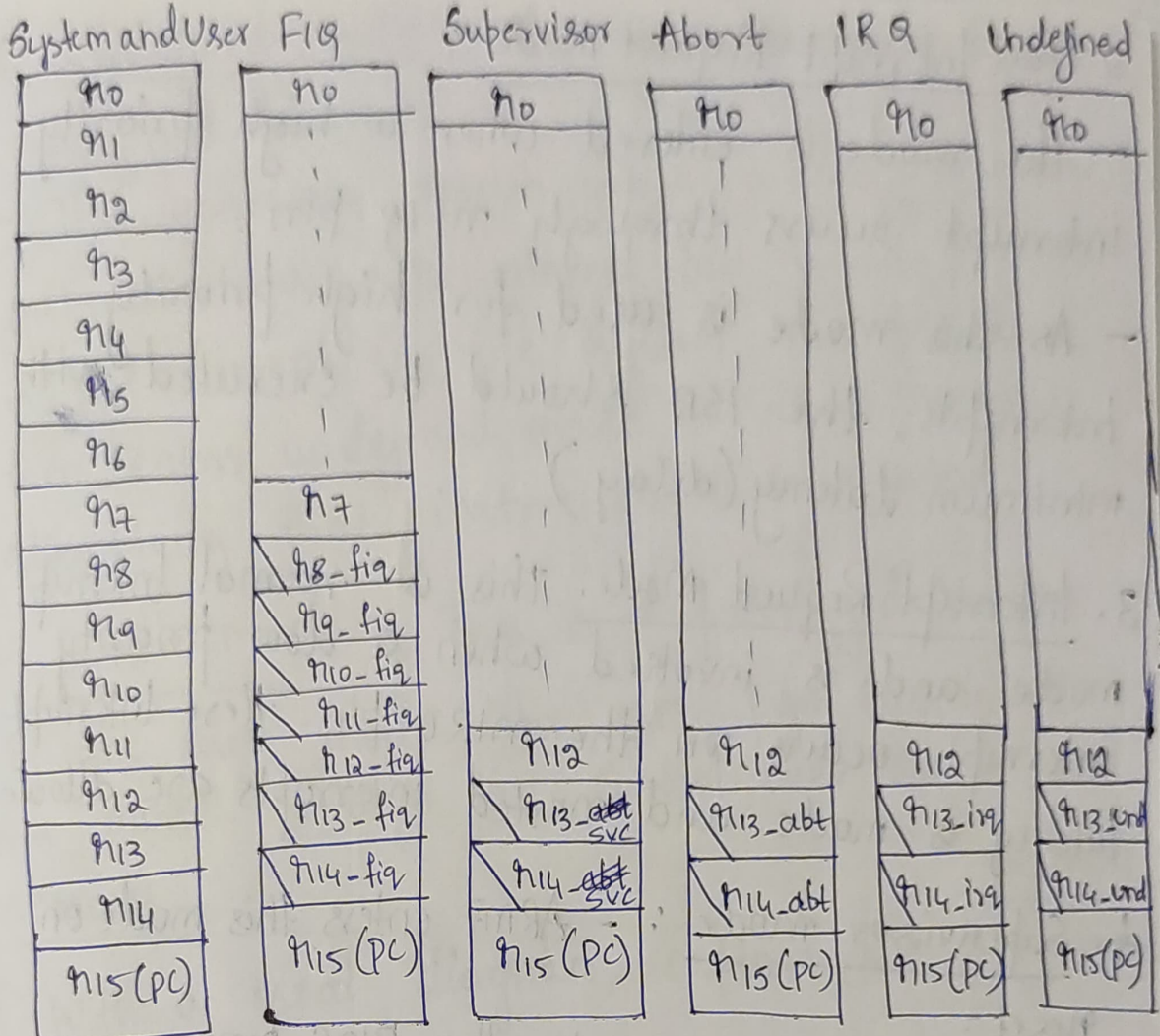
⇒ Each Instruction can be viewed as performing a defined transformation of the states.

- Visible registers.
- Invisible registers
- System memory.
- User memory.

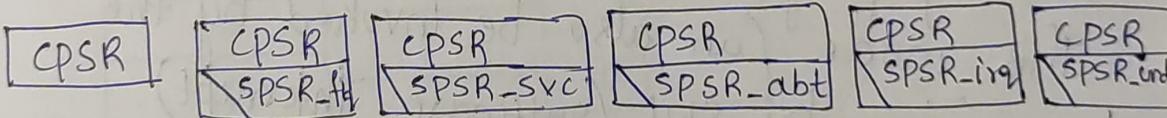
⇒ Processor Modes:

- ARM has seven basic operating modes.
- Mode changes by software control or external interrupts.

CPSR[4:0]	Mode	Use	Registers
10000	User	Normal user code	User
10001	FIQ	Processing fast Interrupts	_fiq
10010	IRQ	Processing std interrupts	_irq
10011	SVC	Processing sw interrupts	_svc
10111	Abort	Processing memory faults	_abt
11011	Undef	Handling undefined instruction traps	_und
11111	System	Running privileged OS	user



ARM - State program Status registers



Δ = banked registers.

⇒ Processor modes: There are 7 modes of operations in ARM7.

1. User mode - This is the normal mode in which user programs are executed.
   - It is the only non-privileged mode.
   - It has limited access to memory, I/O components & flags.

2. Fast Interrupt Request Mode:
- The mode is entered when a high priority interrupt occurs through nFIQ pin.
- As this mode is used for high priority interrupts, the ISR should be executed with minimum latency (delay)

3. Interrupt Request Mode: this is normal interrupt mode and is invoked when a low priority interrupt occurs on the nIRQ pin. Here interrupt latency is mode and nested interrupts are allowed

4. Supervisor mode: - ARM7 enters this mode on reset.
- It is used to execute the BIOS program (Booting program).
- This mode can also be invoked by the programmer by SWI (Software Interrupt).

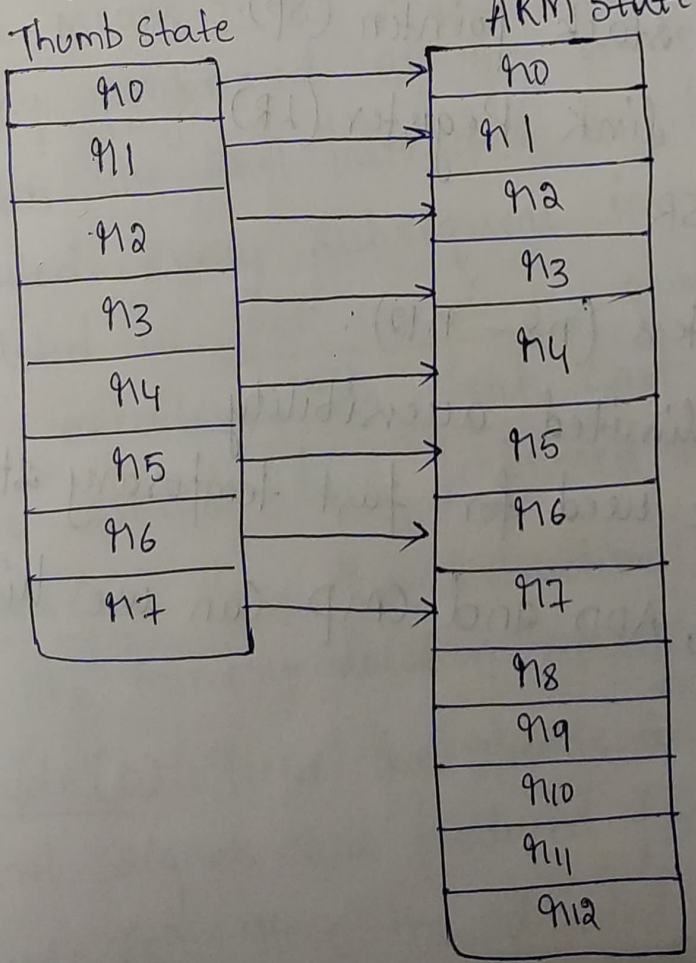
5. Abort mode: This mode is entered when an unsuccesful attempt is made to access a memory location.
- Due to protection mechanism some locations are not accessible in some of the modes.
- when an attempt is made to access such a location, the processor enters abort mode & the program that tried to access this location is aborted.

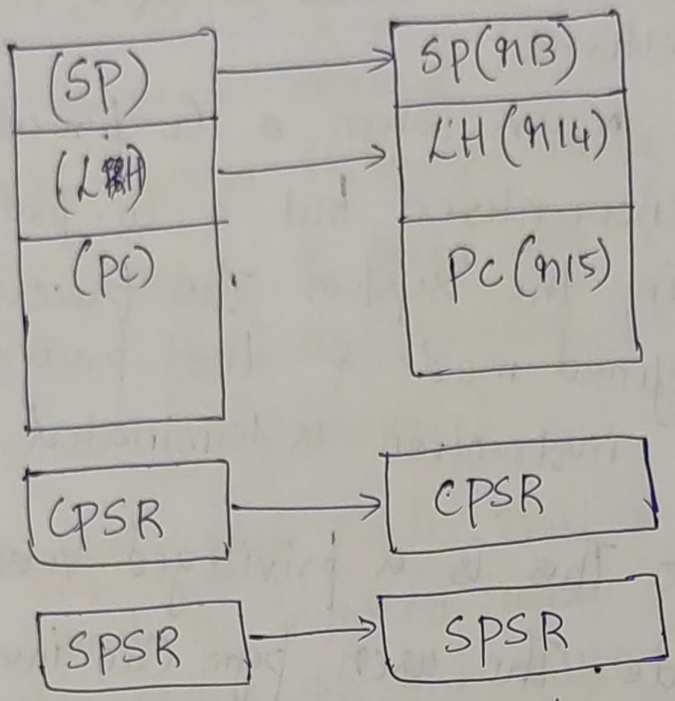
6. Undefined mode - This mode is used to enter undefined Instructions.
- This generally occurs when a Co-processor instruction is encountered but a Co-processor is not present in the system. The processor then enters undefined mode & the program attempting that instruction is terminated.

7. System mode - This is a privileged Version of the user mode. The user pgm can invoke this mode to gain full control over CPSR and memory.

08|12|21

(5) With a neat diagram, explain the programming model for THUMB.





⇒ In Thumb state the programmer has direct access to:

→ eight general registers, R0 - R7

→ R15; the PC

→ R13: a stack pointer (SP)

→ R14: a link Register (LR)

→ The CPSR

⇒ High registers (R8 – R12):

→ It has limited accessibility

→ It can be used for fast temporary storage

→ only MOV, ADD and CMP can use high registers.

⑥ With a neat diagram, explain 3-stage pipeline of ARM.

⇒ ARM 7 Pipelining:-

→ Pipelining means overlapping various stages of an instruction cycle to improve processor performance.

→ ARM 7 implements a 3 stage pipelining:-

- The instruction cycle is divided into 3 distinct stages of Fetch, Decode & ~~Encode~~ Execute.
- So when one instruction is being executed, the next one is being decoded and subsequently next one is being fetched.

→ This means three different Instructions are being performed at the same time.

→ After the first Instruction has been Completely executed, every subsequent instruction gets Completed in just one Cycle, which is needed for execution. The fetch and decode operations have already been performed in the background

→ Fetch (f): Here the instruction will be fetched from the memory location specified by PC.

Decode (D): Here the opcode is decoded and control signals are produced for execution.

Execute (E): Here the decoded Instruction is executed.

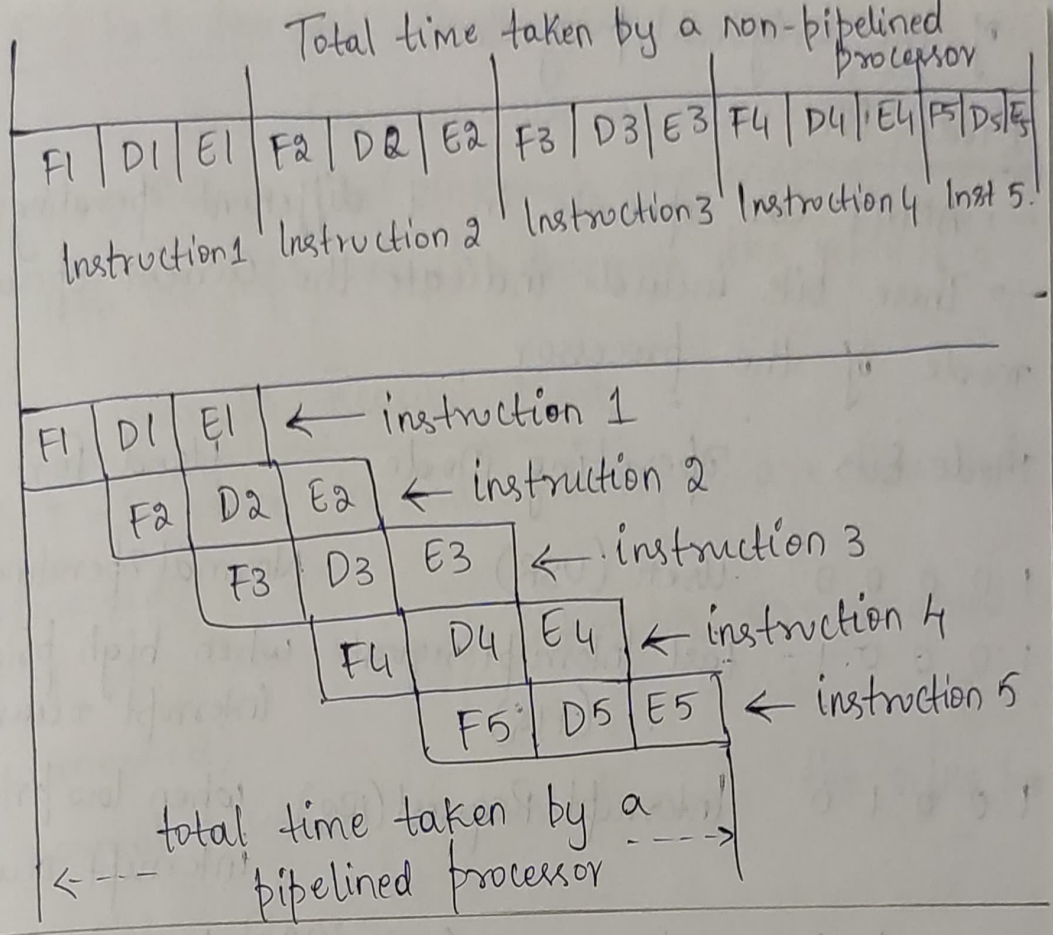
→ ARM7 being a RISC processor, performs superior pipelining due to the following factors:

1. As all instructions are of 32-bits, they take exactly 1 cycle to fetch through the 32-bit data bus.
2. As all instructions perform 32-bit data operations the execution time of almost all instructions is the same that is 1 Cycle.
3. As it follows "load-store" instruction set model, data operations can only be performed on registers.
4. So execution of these instructions does not involve memory access and hence does not interfere with the fetching process.
5. Only load or store instructions need the system bus for execution and hence fetching has to stop during these instructions.

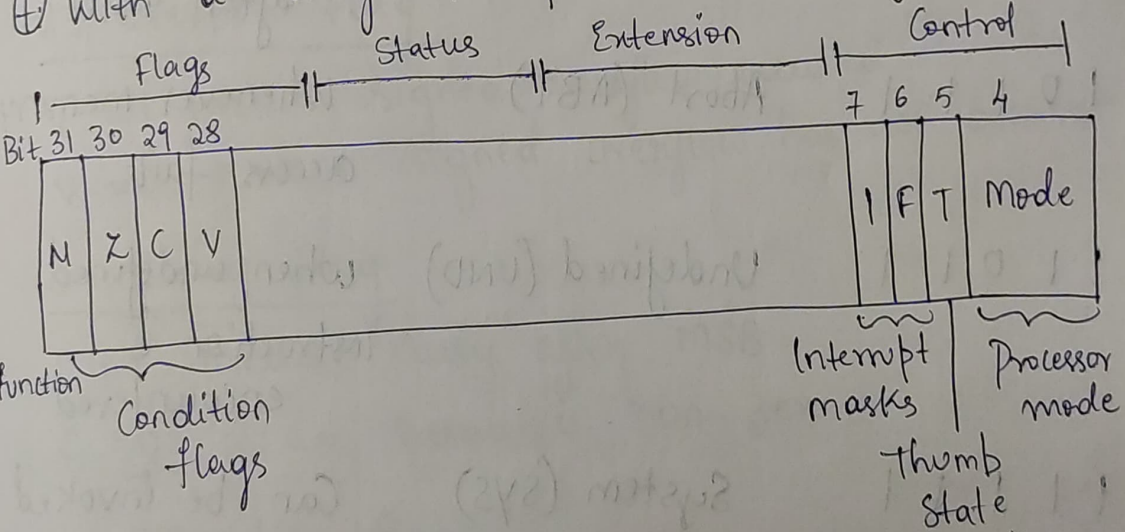
→ Pipelining fails during a branch as it assumes that a program is always executed in a sequential manner.

→ As this is a 3-stage pipeline, PC always points two instructions ahead of the one being currently executed.

→ Higher Versions of ARM processors use deeper pipelines.



④ With a diagram, explain CPSR Register.



CPSR (Current Program Status Register) holds the current status of the program.

When an exception/interrupt occurs, the value of CPSR is saved into SPSR (Saved Program Status Register) before invoking the ISR. While returning to main program, the value of SPSR is put back into CPSR to restore the previous state

of the original program.

Mode Bits

→ ARM7 can operate in 7 different operating modes.
→ These bits include indicate the current operating mode of the processor.

Mode Bits	Operating Mode	Used for
10000	User (USR)	Normal operating mode
10001	Fast Interrupt request (FIQ)	when high priority interrupt occurs.
10010	Interrupt Request (IRQ)	when low priority interrupt occurs
10011	Supervisor (SVC)	default mode after reset.
10111	Abort (ABT)	whenever memory access fails
11011	Undefined (UND)	when undefined instruction is encountered
11111	System (SYS)	Can be invoked by programmer

1. T: Thumb State

If T = 1, then processor is thumb state.
If T = 0, then processor is in normal ARM7 state.
Thumb state is a special state when "Thumb" instruction set, having 6-bit instructions are used.

2. F : Fast Interrupt Mask

If F=1, then Fast Interrupts are disabled (masked)
If F=0, then Fast Interrupts are enabled (unmasked)
A fast Interrupt occurs through the nFIQ pin.

3. I : Interrupt Request Mask

If I=1, then normal interrupts are disabled (masked).
If I=0, then normal interrupts are enabled (unmasked)
A normal Interrupt occurs through the nIRQ pin.

⇒ Condition Flags :-

V : overflow flag

V=1 means signed overflow occured.
V=0 means signed overflow has not occured.

C : Carry Flag

C=1 means Carry after MSB.
C=0 means result is non-zero.

N : Negative flag

If N=1, result is negative
N=0, result is positive.

(8) Explain 7 different modes of ARM.

1. User Mode:

→ This is the normal mode in which all user programs are executed. It is the only non-privileged mode.

→ It has limited access to memory, I/O Components and flags.

→ All the other modes can be entered through different kinds of exceptions (interrupts).

2. Fast Interrupt Request mode:

→ This mode is entered when a high priority interrupt occurs through nFIQ pin.

→ As this mode is used for high priority interrupts, the ISR should be executed with minimum latency (delay).

→ Normally when we begin an ISR the original value of all GPR's of the parent program must be saved and before completion of the ISR these values must be restored into the GPR's for proper continuation of the main pgm.

3. Interrupt Request mode:

→ This is a normal Interrupt mode and is invoked when a low priority interrupt occurs on the nIRQ pin. Here Interrupt latency is more & nested interrupts are allowed.

4. Supervisor mode:

→ ARM7 enters this mode on reset.

→ It is used to execute the BIOS program (Booting program).

→ This mode can also be invoked by the programmer by SWI (Software Interrupt).

5. Abort mode:

→ This mode is entered when an unsuccessful attempt is made to access a memory location.

→ Due to protection mechanism some locations are not accessible in some of the modes.

→ When an attempt is made to access such a location, the processor enters abort mode and the program that tried to access this location is aborted.

6. Undefined mode:

→ This mode is used to enter undefined instructions. This generally occurs when a co-processor instruction is encountered but a co-processor is not present in the system.

7. System mode:

→ This is privileged version of the user mode.

→ The user program can invoke this mode to gain full control over CPSR and memory.

(9) Explain the nomenclature in ARM.

⇒ ARM7: 32-bit Advanced RISC Machine

ARM7TDMI

T - Thumb architecture extension.

- Two separate instruction sets, 32-bit ARM instructions and 16-bit Thumb Instructio

D - Debug extension

M - Enhanced Multiplier.

I - Embedded ICE macrocell extension.

⇒ ARM x y z T D M I E J F S

x - Series

y - MMU

z - Cache

T - Thumb

D - Debugger

M - Multiplier

I - Embedded In-Circuit Emulator (ICE) macrocell.

E - Enhanced ~~DSP circuit~~ Instruction for DSP

J - JAVA acceleration by Jazelle.

F - floating point

S - Synthesizable Version.

(10) What is JTAG, explain the JTAG state diagram

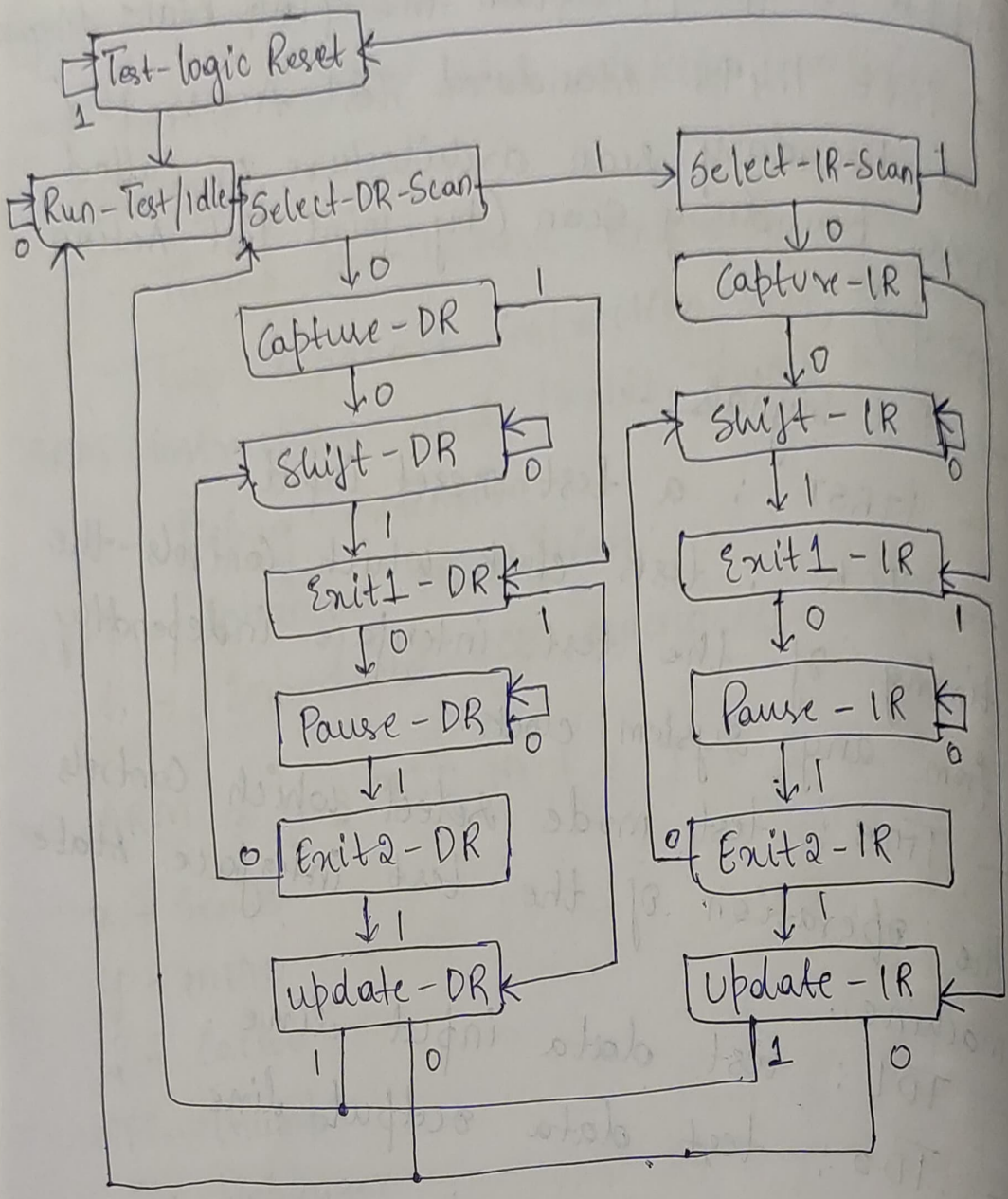
⇒ IEEE 1149, standard Test Access Port and boundary scan architecture or called JTAG boundary Scan (by joint Test Action Group)

⇒ Test Signals :-

- TRST : a test reset input
- TCK : test clock which controls the timing of the test interface independtly from any system clock.
- TMS : test mode select which controls the operation of the test interface state machine.
- TDI : test data input line.
- TDO : test data output line.

⇒ The state machine comprises two paths :

- The Data Register (DR) path used for loading instructions.
- The instruction register (IR) Path used for reading/writing data from/to data registers, including the boundary scan register (BSR).



JTAG Test Access Port (TAP) State Machine.

(11) What is Single Tasking?

⇒ Single Tasking, is the practice of dedicating oneself to a given task and minimizing potential interruptions until the task is completed or a significant period of time has elapsed.

→ In the Single-Tasking, only one task is performed at a time.

→ Microcontrollers are known as Computer on chip. They are designed to perform a single task.

---

(12) What is MMU? Why MMU is required? Give example of MMU system.

⟹ The memory can be defined as a collection of data in a specific format.

→ It is used to store instructions & processed data. The memory comprises a large array or group of words or bytes, each with its own location.

→ The primary motive of a computer is to execute programs. These programs along with the information by access should be in the main memory during execution. The CPU fetches the instruction from memory according to the value of the program counter.

→ Main memory is central to the operation of a Computer. Main memory is a large array of words & bytes, ranging in size from hundreds to thousands to billions. Main memory is a repository of rapidly available information shared by the CPU & I/O devices.

→ Main memory is the place where programs & information are kept when the processor is effectively utilizing them. Main memory is associated with the processor is fast.

→ Main memory is also known as RAM. This memory is a Volatile memory.

*. Memory management:-

→ In a multiprogramming Computer, the OS resides in a part of memory and rest is used by multiple processes. The task of subdividing the memory among different processes is called memory management.

→ Memory management is a method in OS to manage operations b/w main memory & disk during process execution.

*. Why memory management is required?

→ Allocate & de-allocate the memory before & after the process execution

→ To keep track of used memory space by processes.

→ To minimize fragmentation issues.

→ To proper utilization of main memory.

→ To maintain data integrity while executing of process.

Eg:- IBM System/360 model 67, IBM System 370

(13) What is Endianess ? List the types. Give examples.

⇒ In Computing, endianness is the order or sequence of bytes of a word of digital data in computer memory.

→ Endianess is primarily expressed as big endian (BE) or little-endian (LE). ⇒ types

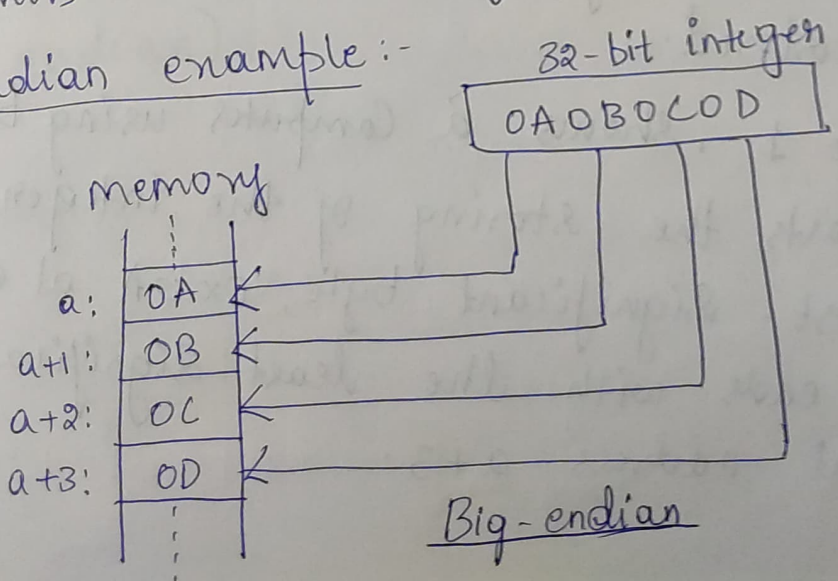
→ A big-endian system stores the most significant byte of a word at the smallest memory address and the least significant byte at the ~~smallest~~ largest memory address.

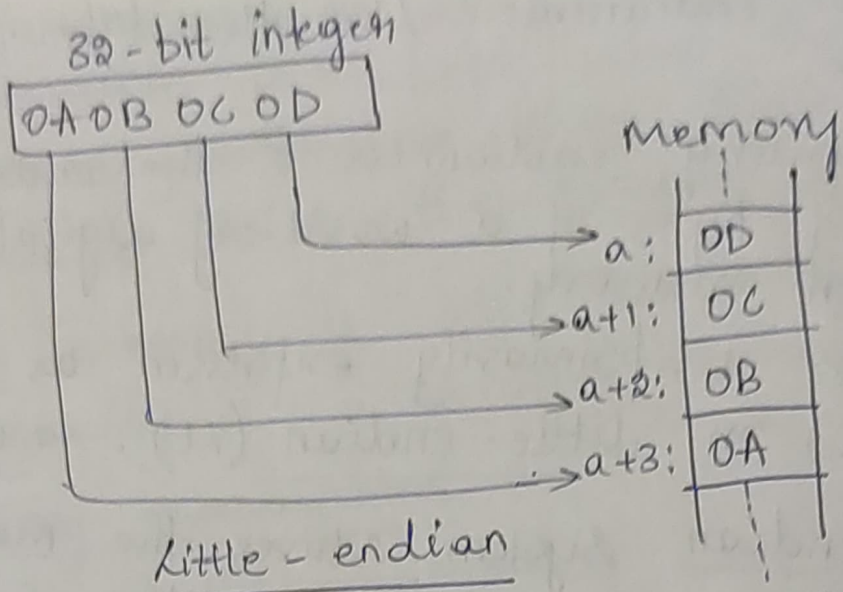
→ A little-endian system, in Contrast stores the least-significant byte at the smallest address.

→ Endianness may also be used to describe the order in which the bits are transmitted over a Communication channel.

Eg. big-endian in a Communications channel transmits the most significant bit first.

→ Endian example :-





Little - endian

→ The 2 diagrams show how two Computers using different endianness store a 32-bit integer with the value of 0x0A0B0C0D.

→ In both cases, the integer is broken into four bytes, 0x0A, 0x0B, 0x0C, 0x0D & the bytes are stored in four sequential byte locations in memory.

→ Starting with the memory location with address a, then a+1, a+2, a+3.

→ the difference b/w big & little endian is the order of the four bytes of the integer being stored.

→ figure 1, shows a Computer using big-endian. This starts the storing of the integer with the most-significant byte, 0x0A at address a and ends with the least significant byte 0x0D at address a+3.

→ figure 2, Computes using little-endian. This starts the storing of the integer with the least-significant byte, 0x0D, at address a, and ends with the most-significant byte, 0x0A at address a+3.

---

(14) Write a C program to find the endianness of given number.

⇒
```
#include <stdio.h>
int main()
{
  unsigned int x = 0x76543210;
  char *c = (char*);
  if (*c == 0x10)
  {
    Printf ("underlying architecture is little endian.");
  }
else
  {
    Printf ("Underlying architecture is big endian);
  }
return 0;
}
```

(15) Explain a) Bit b) Byte c) Nibble d) Half-word e) word.

⇒

a) Bit :- A bit is the smallest unit of information that can be stored in a Computer. Bits in Computer are grouped to form a longer unit of information.
→ A bit has single binary value, either 0 or 1.

b) Byte :- A byte is a Combination of eight bits. Eight bits represent a character and is called a byte.

c) Nibble :- A nibble is a Combination of four bits, in other words a nibble is half a byte.

d) Word :- A word is a Combination of 16 bits, 32 bits or 64 bits depending on the Computer. 16 is known as quad word.

Length	name	Example
1	Bit	0
4	nibble	1011
8	Byte	10110101
16	Halfword	1011010110010011

(16) Explain the word Align and half word align in ARM memory.

⇒ Different processor have different definition of words. for 32-bit processors, a word is 32-bit (4 bytes). As the name implies, a half-word is 16-bit (2 bytes) for 8-bit processor word in 8-bit

→ word allignment : The stored address are adjacent and can be divided by 4, the last 2 digits are 00.

→ Half word allignment : That is the stored address which is adjacent & divisible by 2. ie., last bit is zero.

→ ARM architecture requires 32-bit ARM instructions that must be word aligned & stored in memory & 16-bit thumb instructions requires half-word aligned & stored.

→ Therefore in ARM state the value of R15 is always divisible by 4.

→ In Thumb state, the value of R15 is always divisible by 2, which is the lowest bit of the R15 register always 0.

(7) Explain the software tools involved and processing the C source file with a neat diagram.

→ The ~~following~~ processing of C source file involves 4 major steps, preprocessing, Compiling, Assembly, linking.

→ C source files are by Convention named with .c extension and we use gcc comand to Compile C source files.

⇒ Preprocessing :- preprocessing is the first step. The preprocessor obeys Commands that begin with # (also known as directives) by.

- removing comments
- expanding macros
- expanding included files.

→ If a header file such as #include <stdio.h> it will look for stdio.h file & copy the header file in to source code file.

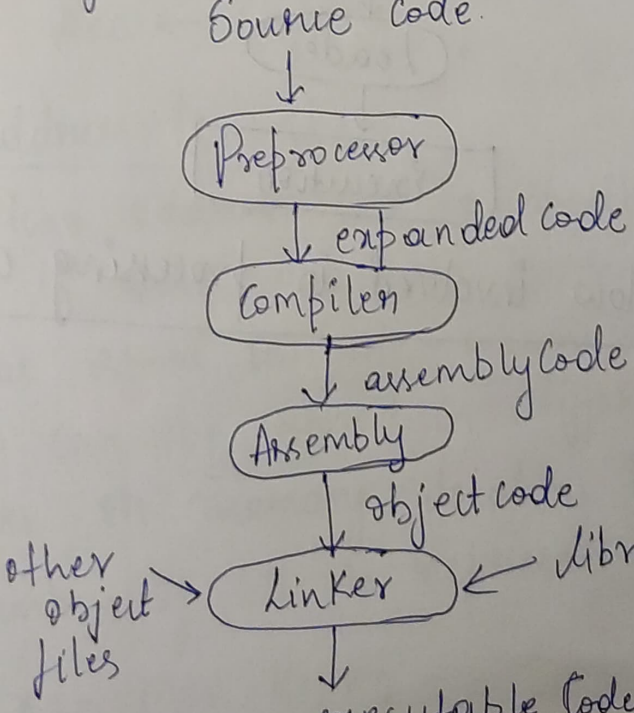
→ Preprocessor also generates macrocode and replaces symbolic Constants defined using #define with their Values.

⇒ Compiling: Compiling is second step it takes the output of the preprocessor and generates assembly language, an intermediate human readable language, specific to processor

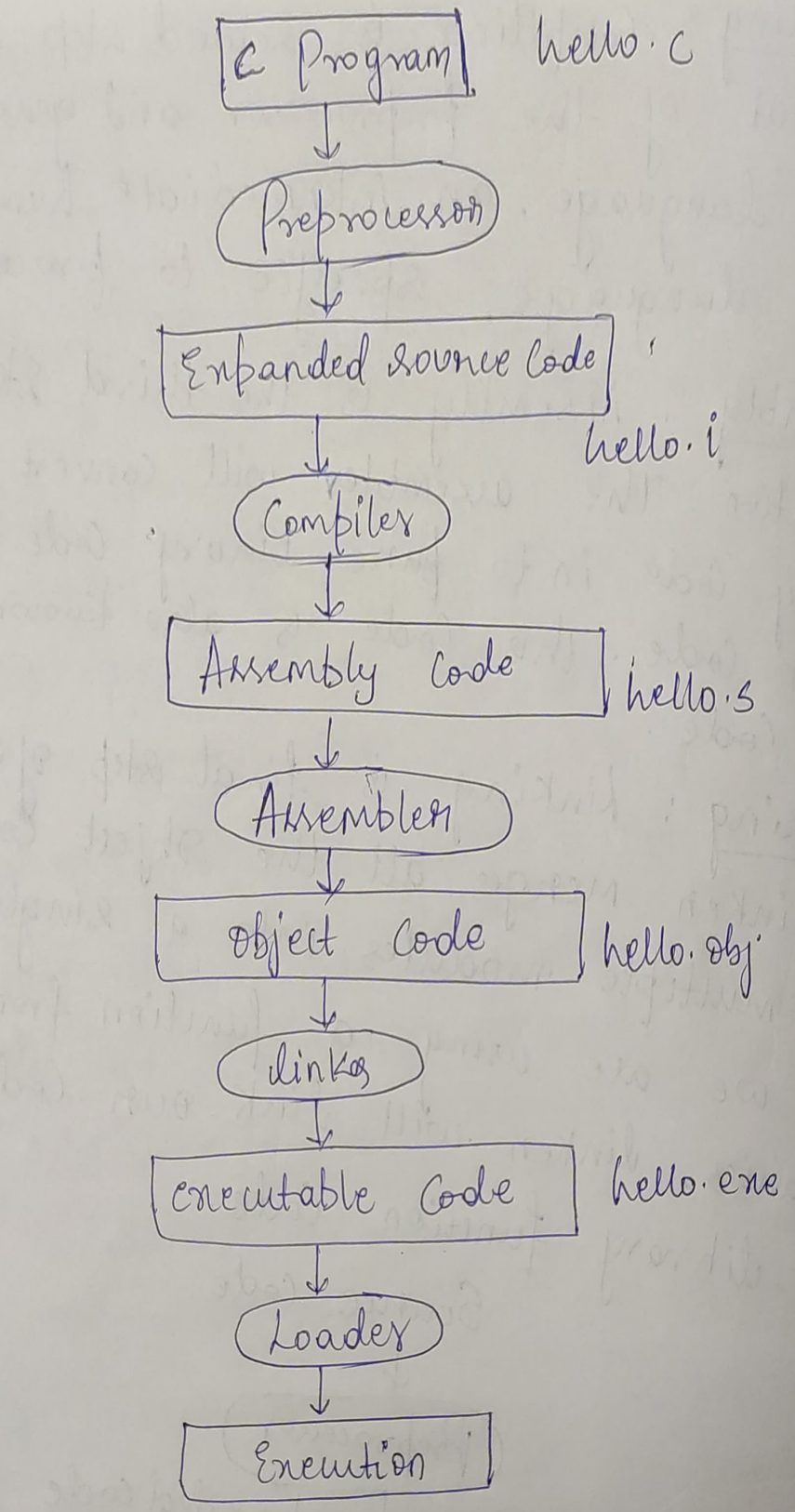
⇒ Assembly: Assembly is the third step of Combilation. The assembler will convert the assembly code into pure binary code (or) machine code. This code is also known as object code.

⇒ Linking: Linking is final step of Compilation. The linker merges all the object code from multiple modules into a single one.

→ If we are using a function from libraries, linker will link our code with that library function code.



a) Software tools involved in processing C source files.



Flow involved in processing c file

(18) Explain the following Addressing modes in ARM. a) Three address b) Two address c) Single address instructions with respect to ARM.

⇒ Processor can execute an instruction only if it is represented in binary sequence.

→ Unique binary sequence pattern must be assigned. This process is called op-code encoding.

a) Single address Instructions:-

→ This uses an implied accumulator register for data manipulation and the other is in the registered memory location. Implied means that the cpu already knows that one operand is in the accumulator so there is no need to specify it

Ex:- LDR addr

Acc ← (addr)

b) Two address Instructions:-

→ Here two address can be specified in the instruction. In single address instruction, the result was stored in the accumulator, here the result can be stored in different locations i.e register or memory location. But require more number of bit to represent the address. Ex:- MOV R1, R2

$R_1 \leftarrow [R_2]$

c) Three address Instructions :-

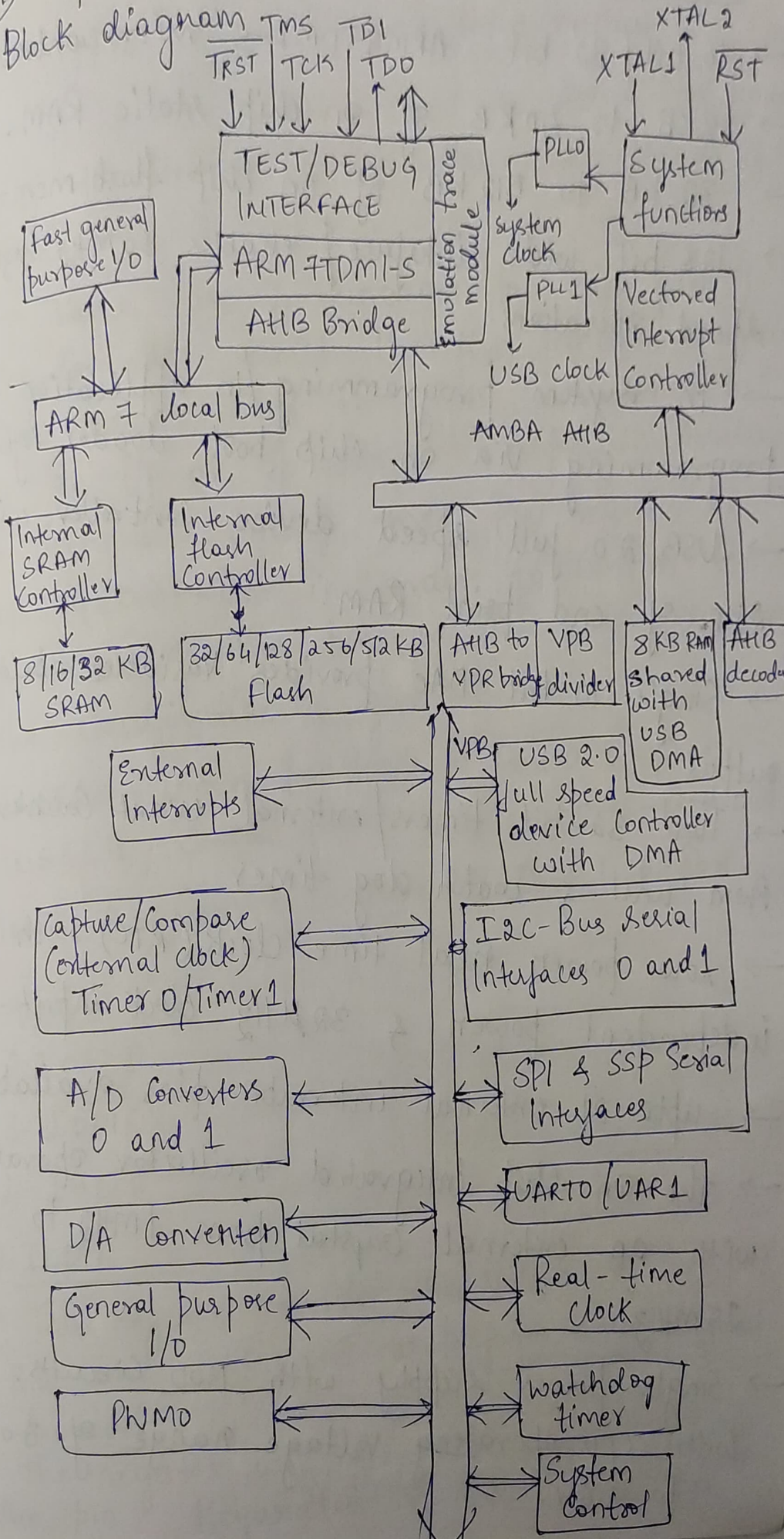
→ This has 3 address field to specify a register or memory location.

→ Program Created are much short in size but number of bits per Instructions increase. Program Created are much short in size but number of bits per instruction increase These instructions make Creation of program much easier but it does not mean that progress will run much faster because, now instruction will only Contain more information but each micro operation will be performed in one cycle only.

Ex :- ADD R_3, R_1, R_2

$R_3 = R_1 + R_2$.

(19) Explain the LPC2148 microcontroller Block diagram



⇒ LPC2148 features :-

→ 16-bit/32-bit ARM7TDMI-S microcontroller
→ 8KB to 40KB of on chip static RAM.
→ 32 KB to 512 KB of on chip flash memory
→ 128 bit wide interface & enables 60MHz high speed operations.
→ In system programming / In application programming via on chip boot loader software
→ USB 2.0 full speed device controller with 2KB of end point RAM.
→ Single 10 bit DAC provides variable analog output
→ Two 32-bit timers / external event Counters, PWM unit & watch dog timer.
→ Low power real time clock (RTC) with independent power & 32MHz clock input.
→ upto 21 external interrupt pins available.
→ the on chip integrated oscillator operates with an external crystal from 1MHz to 25MHz.
→ Single power supply with BOD Circuits. with CPU operating voltage range of 3.0V to 3.6V.

(20) Explain the LPC2148 microcontroller GPIO pins

→ GPIO : General Purpose Input/output.

→ A 32-bit register used to select the function of the pins in which the user needs it to operate.

→ There are 4 functions for each pin of the Controller, which the first function is GPIO.

→ It means that the pin can either act as an i/p or o/p with no specific function.

→ There are totally three PIN register in LPC2148 Controller in order to Control the functions of the pins in the respective ports. The classification is given below:

PINSEL0 - Controls functions of Port 0.0 - Port

PINSEL1 - Controls functions of Port 0.16 - Port 0.31

PINSEL2 - Controls function of Port 1.16 - Port

→ 1. IOPIN
2. IOSET
3. IODIR
4. IOCLR

① IOPIN :-

→ This register provides the value of port pins that are configured to perform only digital functions.

→ The register will give the logic value of the pin regardless of whether the pin

= is configured for input or output or as GPIO or an alternate digital function.

② IOSET :-

→ This register is used to produce a high level output at the port pins configured as GPIO in an output mode.

→ writing 1 produces a high level at the corresponding port pins. writing 0 has no effect.

→ If any pin is configured as an input or a secondary function, writing 1 to the corresponding bit in the IOSET has no effect.

→ Reading the IOSET register returns the value of this register as determined by the previous writes of IOSET register.

③ IODIR :-

→ This word accessible register is used to control the direction of the pins when they are configured as GPIO port pins.

④ IOCLR :-

→ This register is used to produce a low-level output at port pins configured as GPIO in an output mode. writing 1 produces low level at the corresponding port pin & clears the corresponding bit in IOSET register.

→ writing 0 has no effect.

(21) With neat dig, explain Baud rate & Bit rate. Explain the Calculations.

⇒ Baud :- How many times a signal changes per second.

→ Bit :- How many bits can be sent per time unit (usually per second).

→ Bit rate is controlled by baud and number of signal levels.

⇒ Baud rate :-

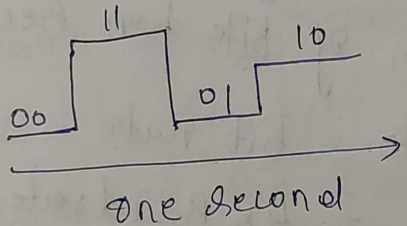
→ No of time lines changed per second.

→ Let Baud rate be 4 (4 changes per second).

→ Let bits per line change be 2.

→ Bit rate = 8 bits per second.

→ Bit rat = x 2 Baud rate in this example.



→ Baud rate defines the switching speed of a signal.

→ Bit rate defines the rate at which information flows across a data link measured in bits/second.

→ For a binary two level signal a data rate of one bit per second is equivalent to one Baud.

→ An analog signal carries 4 bits in each signal unit. If 1000 signal units are sent per second, find the Baud rate & the Bit rate.

1 Bit → 1 Symbol.

Bit rate = Baud rate

$\frac{1000}{1000}$ = Baud rate

Bit rate = 1000 × 4 = 4000 bps

→ If bit rate (or data rate) is "b". Baud rate (or symbol rate) is "s".

→ General formula:-

$b = s \times n$

→ b = DataRate (bits per second)

→ s = Symbol rate (symbols / second)

→ n = number of bits per second.

→ if n=1, Baud rate = Bit rate.

n=4, Bit rate = 4 × Baud rate.

---

(22) With a neat diagram, explain the working features of SPI protocol.

→ the Serial peripheral Interface (SPI) is a synchronous serial Communication Interface specification used for short distance communication.

→ Primarily in embedded system, the interfacing specification was developed by Motorola.

→ SPI Interface bus is Commonly used for interfacing microprocessor or microcontroller with memory like EEPROM, RTC (Real Time clock), ADC (Analog-to-Digital Converters), etc.

⇒ SPI working :-

→ SPI is a standard low-cost and reliable interface developed by Motorola, which is used to interface b/w the microcontroller and its Peripheral ~~interface~~ IC's.

→ Because of its simple interface, flexibility & ease of use, SPI has become a standard.

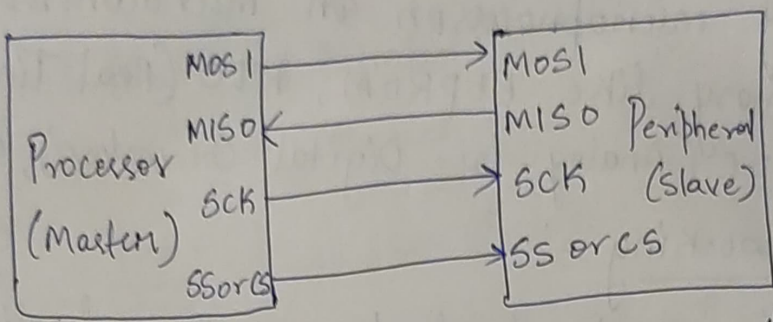
→ In SPI protocol, the devices are Connected in a Master-Slave relationship in a multi-point interface. In this type of interface, one device is considered the Master of the bus (usually a microcontroller) and all the other devices.

→ In SPI protocol, there Can be only one master but many slave devices.

→ SPI bus Consists of 4 signals or pins.

1. Master-Out/Slave-In (MOSI)
2. Master-In/Slave-Out (MISO)
3. Serial clock (SCLK) and
4. Chip Select (CS) or Slave Select (SS).

→ Since, the SPI bus is implemented using 4 signals or wires, it is sometimes called as four wire interface.



→ MOSI : As the name suggests, is the data generated by the Master and received by the slave. Hence, MOSI pins on both the Master and slave are connected together.

→ Master In Slave Out or MISO is the data generated by slave and must be transmitted to Master.

→ To begin SPI Communication, the master must send the clock signal and select the slave by enabling the CS signal.

→ SPI is a full-duplex interface, both Master and slave can send data at the same time via the MOSI and MISO.

→ SPI interface provides the user with flexibility to select the rising or falling edge of the clock to sample and/or shift the data.

(23) In SPI, with neat diagram, explain the CPHA timing and CPOL usage.

→ In SPI, the master can select the clock polarity and clock phase.

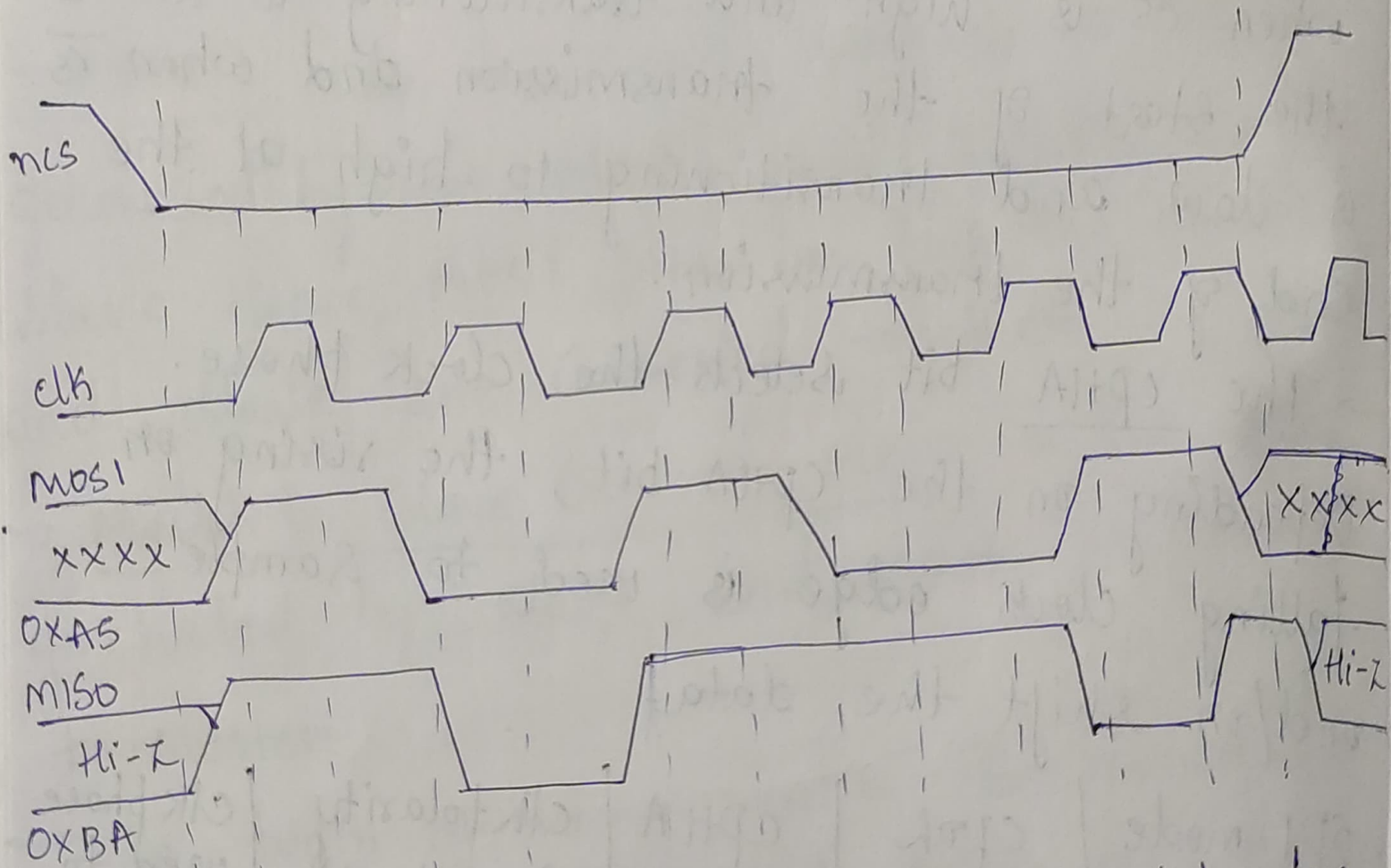
→ The CPOL bit sets the polarity of the clock signal during the idle state.

→ The idle state is defined as the period when $\overline{CS}$ is high and transitioning to low at the start of the transmission and when $\overline{CS}$ is low and transitioning to high at the end of the transmission.

→ The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data.

SPI mode	CPOL	CPHA	clk polarity in idle state	clk phase used to sample or shift data
0	0	0	logic low	Data sampled on rising edge & shifted out on the falling edge.
1	0	1	logic low	Data sampled on the falling edge & shifted out on the rising edge.

2	1	1	logic high	Data sampled on the falling edge & shifted out on the rising edge.
3	1	0	logic high	Data sampled on the rising edge and shifted out on the falling edge.



SPI mode 0, CPOL=0, CPHA=0, clk idle state = low,

(24) with a neat diagram, explain the features of I2C and its working.

→ I2C bus is a two wire serial interface originally developed by the Phillips Corporation

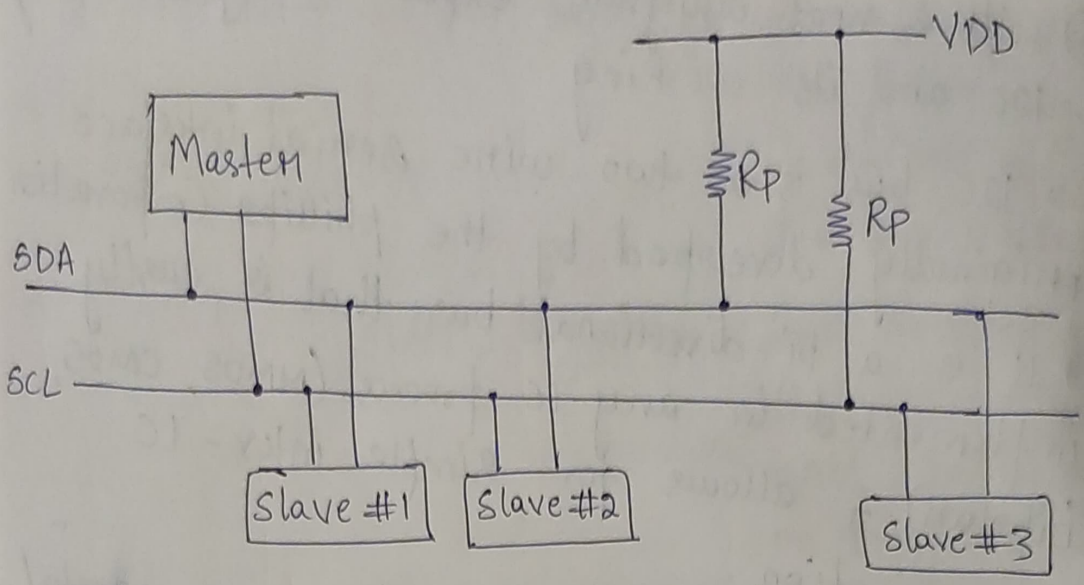
→ It is a bi-directional bus that is easily implemented in any IC process (NMOS, CMOS, bipolar) & allows for simple inter-IC communication.

→ I2C communication protocol follows a master/slave hierarchy, wherein the master is defined as the device that clocks the bus, addresses slaves and writes on reads data to and from registers in the slaves.

→ The slaves are devices that respond only when interrogated by the master, through their unique address.

→ I2C uses only 2 bidirectional lines, Serial Data Line (SDA) and Serial clock line (SCL).

→ I2C compatible devices connect to the bus with open collector or open drain pins which pull the line LOW. When there is no transmission of data, the I2C bus lines are passively pulled high.



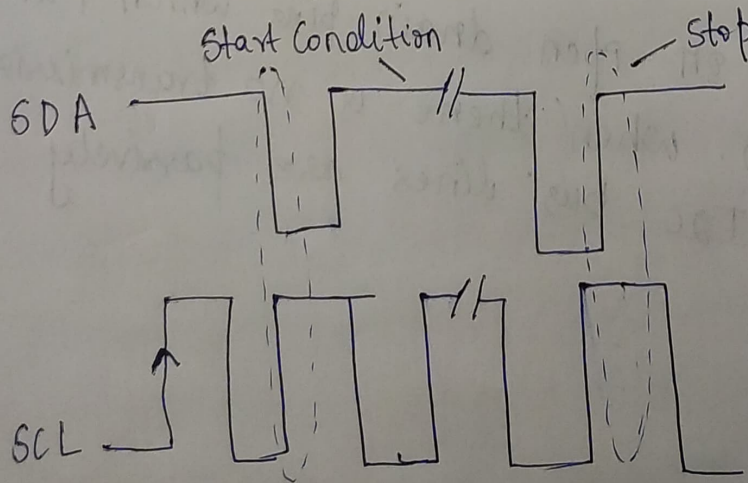
Generalized I2C Connection Diagram

→ The I2C bus can support multiple devices, both SLAVE and MASTER and the only limitation is the capacitance on the bus and the address space as more devices are added.

⇒ Data Transmission Protocol :-

→ I2C data packets are arranged in 8-bit bytes comprising slave address, register number & data to be transferred.

→ Transmission over the bus is either a read or write operation.



Start Condition and Stop Condition Transitions

⇒ Acknowledge and Not Acknowledge Bits (ACK/NACK)

→ As a form of feedback, after every byte transmission the receiving device sends an Acknowledge or Not Acknowledge bit.

→ An Acknowledge bit is generated by the receiver by holding the SDA line low during a HIGH SCL period, ~~while~~

→ A Not Acknowledge bit is generated when the receiver leaves the SDA line passively pulled HIGH and does not respond in any way. This fact implies that in response to an address byte, all unmatched SLAVES send a Not Acknowledge bit by not responding.

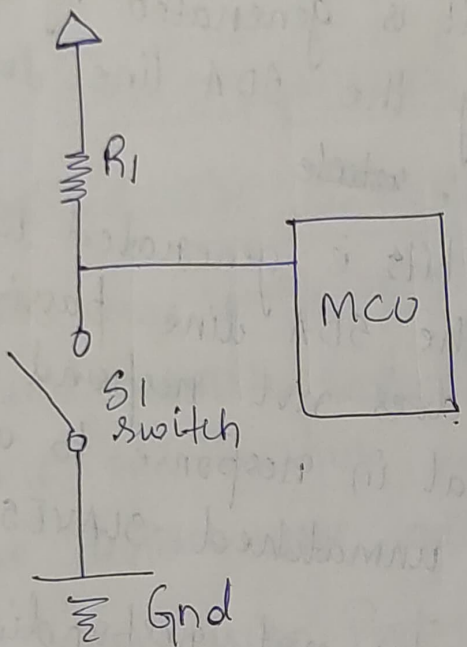
(25) With a neat diagram, explain the pullup and pulldown resistor.

⇒ Pull-up resistors are resistors used in logic circuits to ensure a well-defined logical level at a pin under all conditions.

→ Pull-up resistors are resistors which are used to ensure a wire is pulled to a high logical level in the absence of an input signal.

→ Digital logic circuits have 3 logic states: high, low & floating.

→ high-impedance state occurs when the pin is not pulled to a high or low logic level but is left "floating" instead.

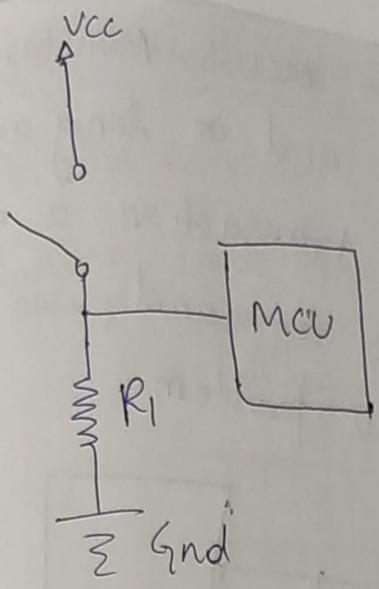


Pull-up resistor Circuit

⇒ Pull-down resistors work in the same manner as pull-up resistors, except that they pull the pin to a logic low will.

→ They are connected b/w ground & the appropriate pin on a device.

→ The pull-down resistor must have a larger ~~resistor~~ resistance than the impedance of the logic circuit.



Pull-down resistor.

(26) with a neat diagram, explain the Concept of arbitration in I2C.

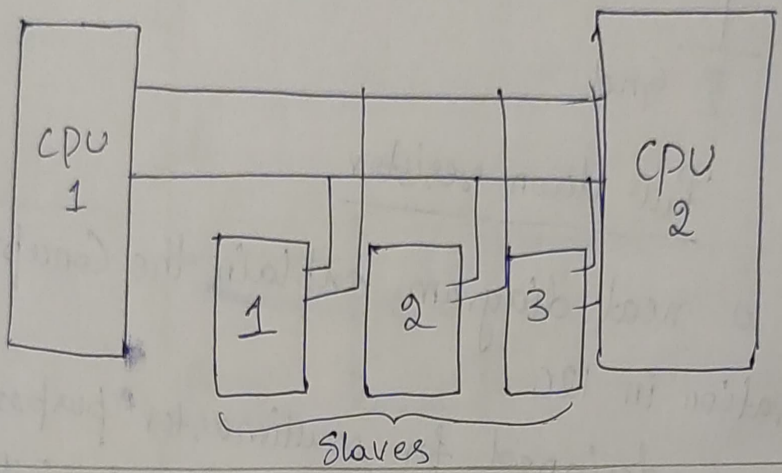
⇒ I2C is designed for multimaster purpose that means, more than one device can initiate transfer.

→ Bus arbitration occurs when two or more Master start a transfer at the same time.

→ the I2C bus was originally developed as a multimaster bus. This means that more than one device initiating transfer can be active in the system.

→ when using only one master on the bus there is no real risk of captured data, exept if a slave device is malfunctioning or if there is a fault condition involving in the SDA/SCL bus.

→ As long as the two MCU's monitor is going on the bus and as long as are aware that a transaction is g on because that lack command was STOP; these occurs no problem.



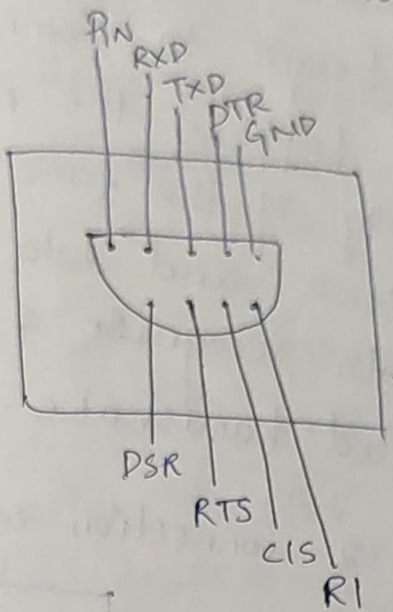
(27) What is clock stretching? Explain clock stret in I2C.

⇒ Clock stretching allows an I2C slave device to force the master device into a wait state. A slave device may perform clock stretching when it needs more time to manage data such as store received data or prepare the transmit another byte of data.

→ Clock stretching in I2C devices can slowdown communication by stretching SCL :- during an SCL low phase any I2C device on the bus may additionally hold down SCL to prevent it to rise again enabling them to slow down the SCL clock rate or to stop I2C communication for a while.

This is also referred to as clock synchronization

(8) Explain the working of DB9 pins and handshaking with the modem



Pin	Signal	Signal name	DTE signal direction
1	DCD	Data Carrier detect	In
2.	RXD	Receiver data	In
3.	TXD	Transmit data	out
4.	DTR	Data terminal ready	out
5.	GND	Ground	—
6.	DSR	Data Set Ready	In
7.	RTS	Request to Send	out
8.	CTS	Clear to Send	In
9.	RI	Ring Indicator	In

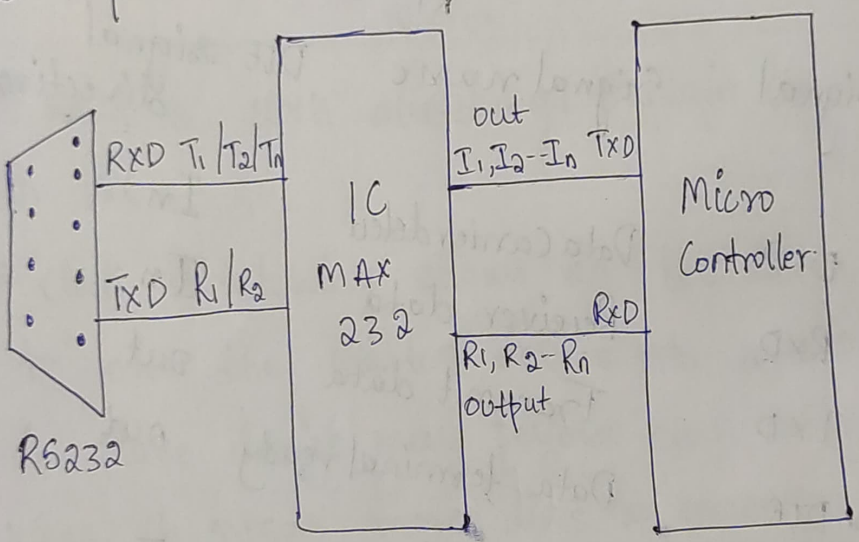
⇒ Handshaking modem :-

→ A modem handshake is what occurs when the receiving modem answer the phone call and the two modems begin to communicate.

→ Before anything happens the modem must evaluate the quality of the time negotiate error control protocols and data comparision that they can both recognise & work. This process is called handshake.

---

(29) Explain the RS232 Connection with a microcontroller.



⇒ Several devices collect data from sensors and send it to another unit like a computer for further processing. Data transfer / communication is generally done in two ways.

→ Serial communication on the other hand uses only one or two data lines transfer, which is generally used for long distance communication.

→ An important parameter considered while interfacing signal port is the Baud rate which is the speed at which data is transmitted serially.

→ Microcontroller can be set to transfer & receive signal data at different baud rate using software instructions.

---

(30) Explain the frame format in UART Communication

⇒ Baud rate - Baud rate in a data transmission refers to the number of symbols transferred per second. A symbol is a group of a fixed number of bits.

→ Data framing - UART transmits data in packets. Each data packet may contain one start bit, 5 to 9 data bits, an optional parity bit and 1 or 2 stop bits

START	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	P_B	STOP

→ UART receives the data from the data bus and this data are being sent by CPU, memory or microcontroller.

→ The data transmission from the data bus to UART is in parallel mode.

→ UART adds the start bit, parity bit and a stop bit to the data received from the data bus.

which creates a data packet.
→ This data packet is serially transmitted to the receiving UART by the transmitter.
→ The receiving pin of the receiver UART reads the data bit by bit. this data is again converted into parallel form at the receiver.
→ Start bit : When there is no data transmission the UART transmission line is held at high voltage to transition acts as the start bit. When the receiving UART delete the high to low voltage transition, it begins reading the data frame at the frequency of baud rate.
→ Data frame : The data bits are usually 5 to 8 in members. If no parity bit is used, it can be 9 bit long. In general case the LSB of the data is transmitted first.
→ Parity bit : The parity bit is used to indicate the change in data during transmission. The reason for change in data is mismatched baud rate.
→ Stop bit : To mark the end of the data present, the sending UART drives the data transmission line from a low to high voltage

(5) Explain the difference between in detail
a) Serial v/s Parallel
b) Analog v/s Digital
c) Synchronous v/s Asynchronous.

a)

Serial	Parallel
→ Data is transmitted bit after the bit in a single line	→ Data is transmitted simultaneously through group of lines
→ Data Congestion takes place	→ No data Congestion
→ Low speed transmission	→ High speed transmission
→ Implementation of Serial links is not an easy task	→ Parallel data links are easily implemented as hardware
→ No Crosstalk problem	→ Crosstalk Creates interface b/w parallel lines
→ The bandwidth of Serial wire is much higher	→ the Bandwidth of parallel wire is much lower

b)

Analog	Digital
→ Transmitted modulated signal is analog in nature	→ Transmitted signal is digital. ie, train of digital pulse.
→ Amplitude frequency or phase variations in the transmitted signal represent the information or message	→ Amplitude width or position of the transmitted pulse is transmitted in the form of code words

→ Noise immunity is poor.	→ Noise immunity is excellent.
→ Cooling is not possible	→ Cooling techniques can be used to detect & correct the error.
→ FDM is used for multiplexing	→ TDM is used for multiplexing.
→ Analog modulation system are AM, FM, PM, PAM, PWM	→ Digital modulation system are PCM, DM, ADM, DPCM.
c) Synchronous	Asynchronous
→ Communicated in real time	
→ Creates interrupt in working	→ Eliminates Interrupt
→ Sends the data & receives data on same clock frequencies	→ Sends & receives data on different clock frequencies.
→ Faster	→ Slower
→ No overhead of extra start & stop bit	→ User Start & Stop bit.
→ Uses constant time interval	→ Uses random or irregular time interval