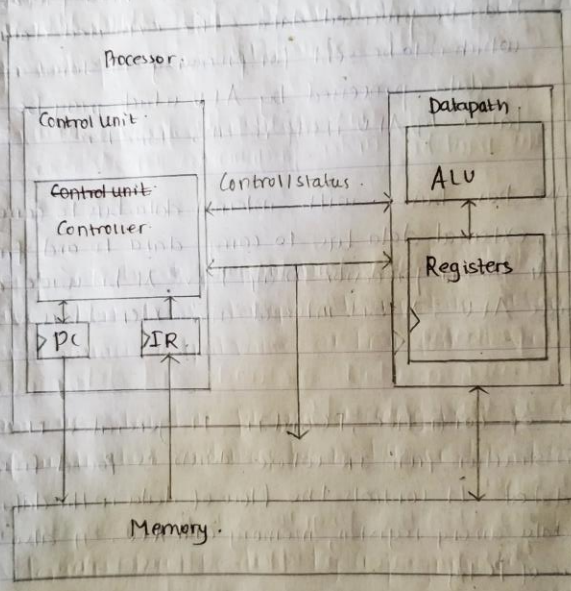
Bipin K C.
Embedded system.
Assignment 3

Software design issues.

1. Explain basic architecture of GPP



The basic architecture of a GPP consists of following units.

1. Datapath.
   It is a circuitry for transforming data. It consists of two units ALU and registers.
   - ALU: It performs operations like addition, subtraction,

logical AND/OR and also generates status signals to indicate particular data condition like generating flags (status, zero, carry etc).

- Registers: They store data which might be need processing or may be processed data. The extracted data from memory which is to be sent for processing is stored in registers. or data processed by ALU which may be required later by ALU itself is also stored here.

Data bus also has buses with internal data bus to carry data within datapath or external data bus to carry data to and from data memory. An N-bit processor may have N-bit wide registers, N-bit wide ALU, N-bit internal and external bus.

Control Unit.

It consists of controller, PC and IR. It generates the control signals to read instruction from IR whose address is found from the PC and PC is updated. It controls the flow of data in the datapath through transfer of data among register, memory and ALU. It also determines the next value of PC by;

- If the instruction is non-branched PC is incremented.
- If the instruction is branched then status signals and IR determine the next address.

The process that occurs in this unit can be listed under following topic.

1. Fetching instruction.
2. Decoding instruction.
3. Fetching operands.

4. Executing instruction in datapath.
5. Storing results.

The state registers are Program Counter(PC) and Instruction Register(IR).

PC: It holds the address of the next instruction. The bit width of PC represents processor's address size.

IR: It holds the fetched instruction.

3. Memory:

Memory ~~const~~ can be a program memory and data memory. The registers have short term storage storing data information which may be input values, output values or transformed values by program.

The program are stored in program memory that ~~is~~ consists of sequence of instructions and may be medium or long term storage.

2. What considerations must be done by a programmer?

- The programmer doesn't need the detailed understanding of the architecture. He just needs to know what kinds of instructions can be executed. He needs to know about program and data memory space.
- The programmer needs to know about registers which is a direct concern for assembly-level programmer. Also the number of registers is to be known.
- The programmer needs to know different level of instructions

Machine level: Here the codes are in binary form.
Assembly level: The processor specific instructions where mnemonics are used.
Structured languages: The instructions are processor independent.

The programmer needs to have knowledge about I/O signals and how to communicate with external signals.

The interrupt which may arise during the process and how to handle interrupts also must be under programmer's consideration.

Define development and target processor.

Development processor:
The processor on which we write and debug our program. This may usually be a PC.

Target processor:
The processor that the program will run on in our embedded system. This processor is often different from the development processor.

Explain different implementation tools for software development.

1. Assembler.
It converts assembly instructions to binary machine instructions.
Assemblers replace op-code and operand mnemonics by binary equivalent. It translates symbolic labels into actual addresses.
For instance, in a jump code (J2) the jump code after checking a condition transfers the main function to a certain location.
Assembler performs one to one mapping of assembly instruction to machine instructions.

2. Compilers.
Compilers convert high level programs into machine programs. Each high level construct may translate to several machine instruction.
Cross Compiler: This compiler runs on one processor but generates code for another processor.
For instance: The program running on PC may generate code for 8085 µP)

3. Linker.
Linker allows to create a program in separately assembled or compiled files. Linker combines machine instructions of user code and instructions from standard library.

5- Describe various verification tools of system development.

1) Debuggers
The programs run on development processor but execute code designed for target processor. It simulates the function of target processor. It allows to evaluate and correct programs in development processor. The techniques supported by debuggers are stepping, breakpoints, watch values.

2) Instruction set Simulators or Virtual Machines.
In such tool the program is tested in development processor thus making design cycle is fast. Since tested in development processor there is less interaction with actual system and its environment thus inaccurate.

3. Emulators:
These are hardware or software that enables one system to behave like another system. It consists of debugger coupled with a board connected to development processor. The board consists of
- a target processor or device similar to target processor.
- support circuitry.

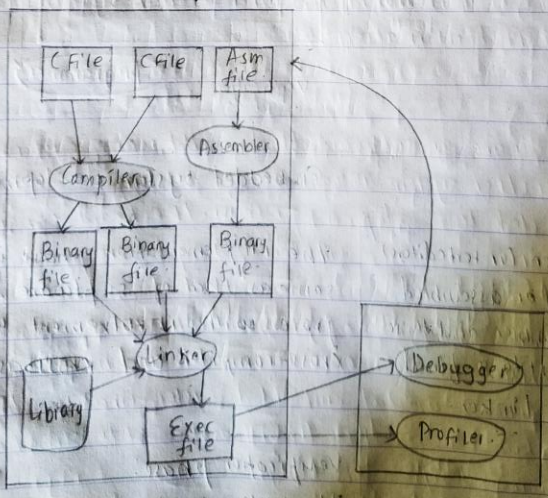
Emulators support debugging of program while it executes on target processor. Thus enables one to control and monitor the program's execution in actual embedded system circuit. It provides accurate testing but code must be downloaded into emulator hardware making design cycle longer.

4. Device programmer.

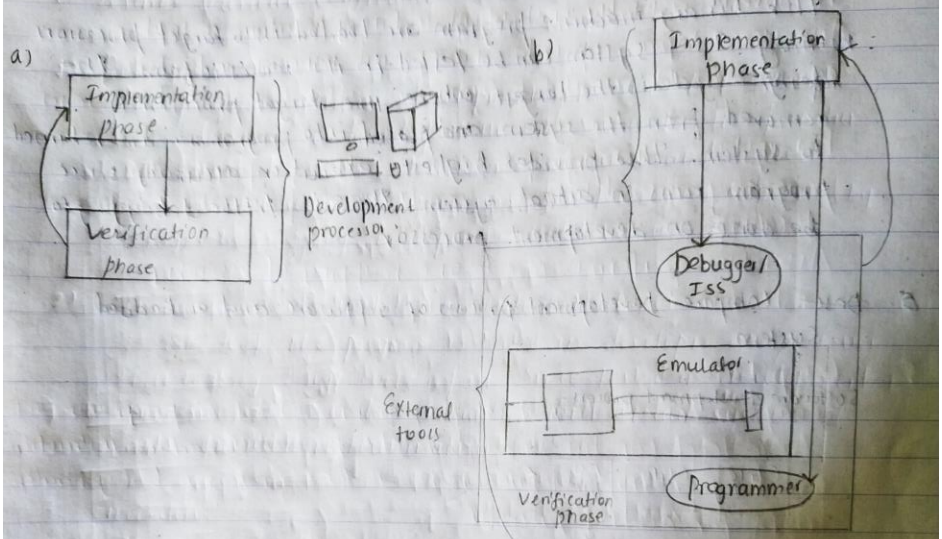
The binary machine program are loaded into target processors memory. The system can be tested in it's realistic form. The design cycle is the longest where the target processor is removed from the system and plated into programmer and returned to system. This provides highest execution accuracy since program runs in actual system. There is little debugging to be done on development processor.

6. ~~Dest~~ Compare development process of software and embedded system.

Software development process.



Embedded system Development process.



Software development	Embedded system development.
Implementation phase: A source code is written (interted/tool) which is then compiled or assembled using compiler or assembler and these files are combined into a final executable file using linker.	The implementation phase is same as that of software development process with involvement of cross-compilers and cross-assemblers.
Verification phase: The executable file are run under the	Verification phase: The system works in conjuction with

command of a debugger.
The program's behaviour is checked by all possible inputs, especially boundary cases.

Other components as well as with environments. The debugging program requires control over time, environment and ability to trace or follow the execution of program.
The verification is done by use of debuggers, emulators or device programmer on a basis of requirement and availability.

Describe about different ASIPs.

ASIP are targeted to a particular domain containing architectural features specific to that domain

i) Microcontrollers.

They perform control-oriented tasks. A microcontroller includes several peripheral devices like timer, ADC, serial communication devices etc.
They result in compact and low power implementation.
Applications.
a) They are used in reading sensors, setting actuator.
b) They deal with small amount of data (in bits).

They have on chip program and data memory. The programmer can access to the chip's pins directly. They also have specialize instructions for bit-manipulation.

ii) Digital signal processors (DSP)

They are optimize for processing large amount of data. which maybe image captured by a camera, voice packet through a network router, audio clip played by an instrument. DSP has numerous register files, memory blocks, multipliers and other arithmetic units. They provide instructions that are central to digital signal processing through filtering and transforming vectors of data. They performs faster arithmetic executions. and also allow parallel execution boosting the performance.

DSP have many peripherals like ADC, DAC, PWM, DMA controllers etc.

iii) less-General ASIP.

These are designed to perform some very domain specific processing while allowing some degree of programmability.
An example of less general ASIP is,
Networking hardware. This is programmable with different network routing, checksum and packet processing protocols.

3. Explain the criteria for selecting a microprocessor.

i) Technical Aspect

We need to know our requirements and select one with desired speed with certain power, size and cost constraints.

ii) Non technical aspects.

We need to check for suitable development environment, prior expertise and licensing.

iii) Clock speed of processor.

~~Even though the clock speed may be higher but instructions per cy~~

Along with the clock speed of processor the instructions per cycle may differ and also the instruction executed per second can be decisive. (For same operation, one processor may require 100 instructions while another may require 200 instructions).

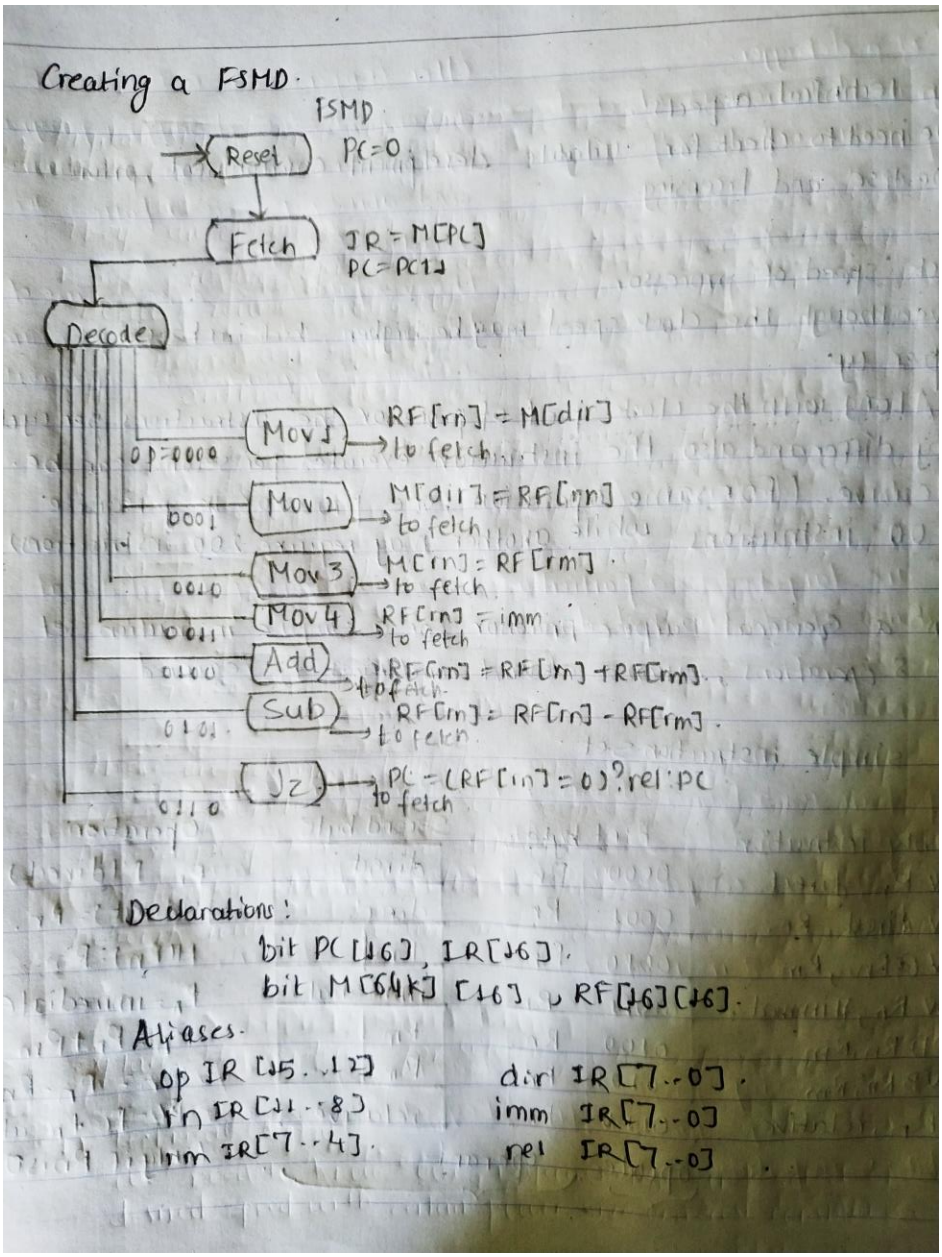
9. Design a General Purpose Processor for a simple instruction set with 8 operations, assume relevant conditions if required.

A simple instruction set

Assembly instruction	First byte		Second byte	Operation
MOV Rn, direct	0000	Rn	direct	Rn = M(direct)
MOV direct, Rn.	0001	Rn.	direct	M(direct) = Rn.
MOV @Rn, Rm	0010	Rn	Rm	M(Rn) = Rm
MOV Rn, #immed.	0011	Rn	immediate.	Rn = immediate
ADD Rn, Rm	0100	Rn.	Rm	Rn = Rn + Rm
SUB Rn, Rm.	0101	Rn	Rm	Rn = Rn - Rm
JZ Rn, relative	0110	Rn	relative.	PC = PC + relati (only if Rn is 0)

opcode | operand

Creating a FSMD.



Declarations:

bit PC[16], IR[16].

bit M[64K][16], RF[16][16].

Aliases.

op IR[15..12]

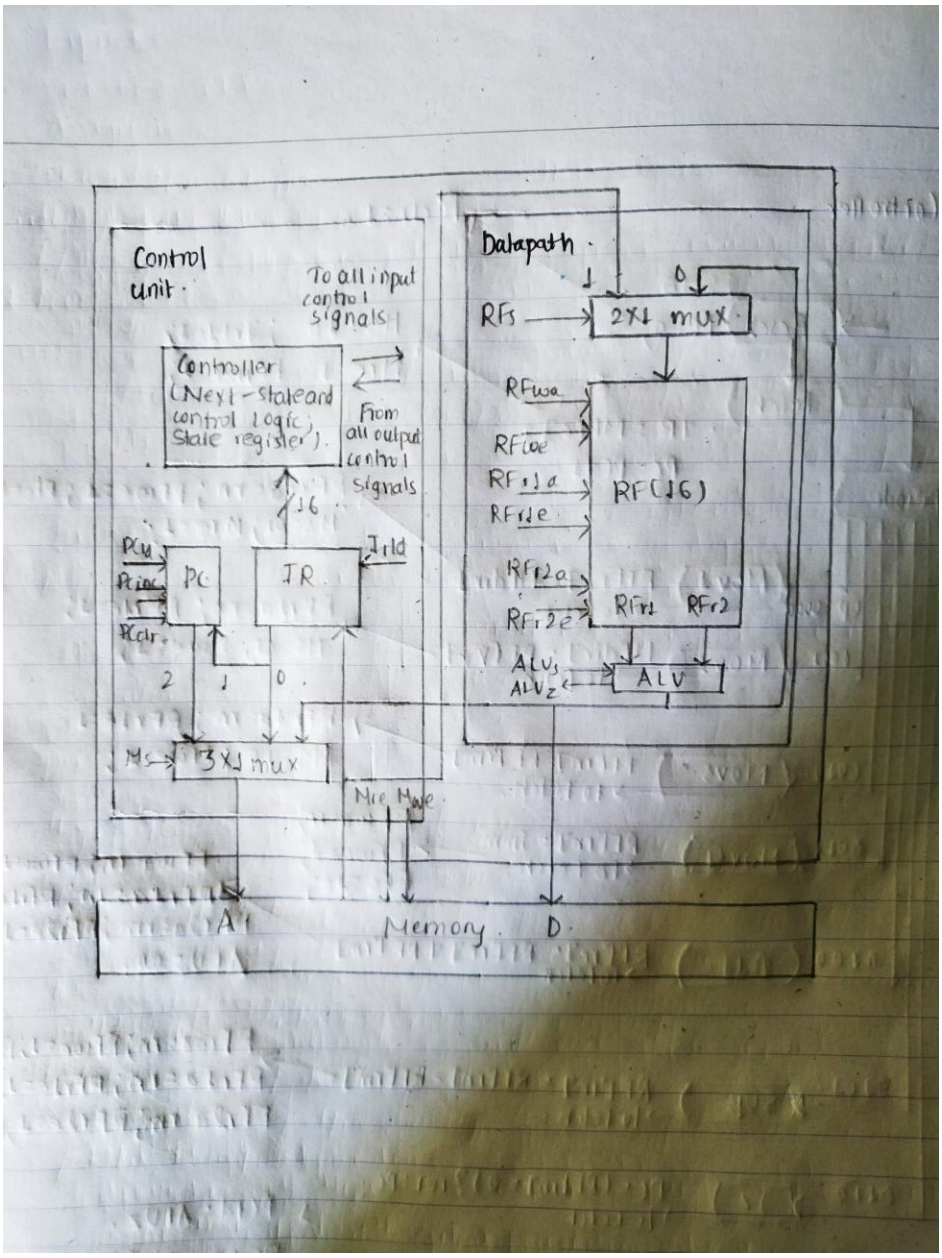
rn IR[11..8]

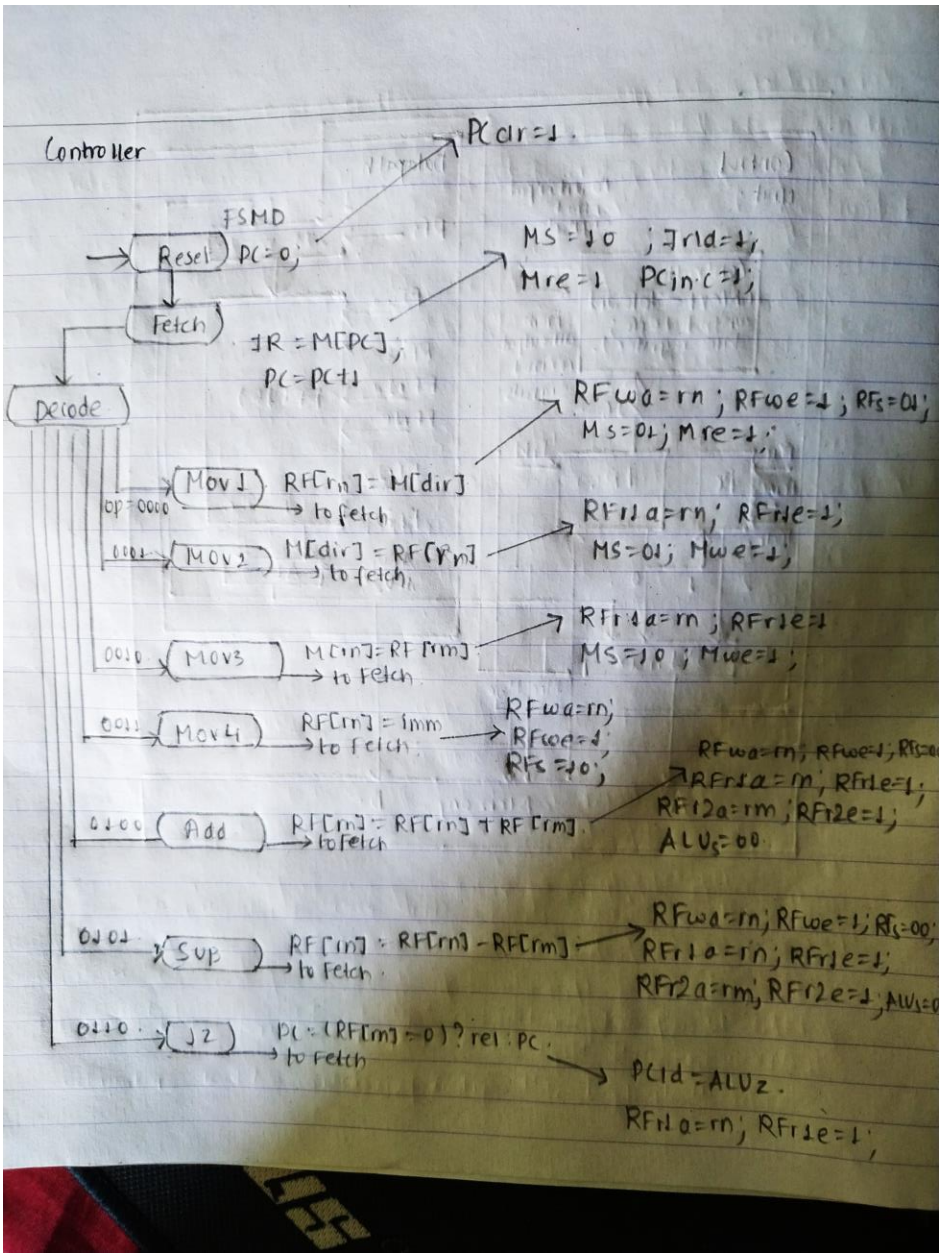
imm IR[7..4].

dir IR[7..0].

imm IR[7..0]

rel IR[7..0]


Control unit
To all input control signals
Controller (Next-state and control logic, State register).
From all output control signals
16
PCld
PCinc
PCclr
PC
IR
IRld
2
1
0
MS
3X1 mux
Mre Mwe
A
Memory
D
Datapath
RFs
2X1 mux
RFwa
RFwe
RFra
RFre
RF(16)
RFrd2a
RFrd2e
RFr1
RFr2
ALUs
ALUz
ALU


Controller
FSMD
Reset
PC=0;
PCclr=1;
Fetch
IR=M[PC];
PC=PC+1
MS=10 ; IRld=1;
Mre=1 PCinc=1;
Decode
Mov1
RF[rm]=M[dir]
to fetch
RFwa=rn; RFwe=1; RFs=01;
MS=01; Mre=1;
op=0000
0001
MOV2
M[dir]=RF[rm]
to fetch;
RFra=rn; RFre=1;
MS=01; Mwe=1;
0010
MOV3
M[rn]=RF[rm]
to Fetch
RFra=rn; RFre=1
MS=10; Mwe=1;
0011
MOV4
RF[rn]=imm
to fetch
RFwa=rn;
RFwe=1;
RFs=10;
0100
Add
RF[rn]=RF[rn]+RF[rm]
to fetch
RFwa=rn; RFwe=1; RFs=00;
RFra=rn; RFre=1;
RFr2a=rm; RFr2e=1;
ALUs=00
0101
Sub
RF[rn]=RF[rn]-RF[rm]
to Fetch
RFwa=rn; RFwe=1; RFs=00;
RFra=rn; RFre=1;
RFr2a=rm; RFr2e=1; ALUs=01
0110
JZ
PC=(RF[rn]=0)?rel:PC;
to Fetch
PCld=ALUz.
RFra=rn; RFre=1;