Mid Semester Examination CS 204
1.5 Hours, Maximum Marks 50

- **ALL PARTS of a question MUST be answered in sequence**
- **Neat drawing is mandatory. Striking off and cluttering of drawings may lead to non-evaluation/non-marking of such answers.**
- **No doubts will be entertained. Please write yours assumptions (if you feel inconsistency regarding the questions).**

Question 1 ----------[Marks 5- bus Arch, Marks- 20 for the signal sequence and explanation/illustration]

Consider an accumulator based processor with 8-bit data bus and 8-bit address bus. Apart from accumulator (ACC), it has 6 other 8-bit general purpose registers (A to H). It has also a condition code register for flag as SIGN, ZERO, CARRY, OVERFLOW, etc. Also consider the special purpose register MAR, MBR, PC and IR, SP (stack pointer).

Write the **Sequence of Signals and Comments** required for the **fetch-decode-execute** steps of the instructions given below. These data movements (step-wise) must be illustrated in a **one-bus CPU architecture**. So, you need to first design **a single bus Architecture**.

A) ADD src
The length of the instruction is one byte. It adds the contents of src to the accumulator and store the result in accumulator. Consider register C as src.

B) JMPN dst
The length of the instruction is one byte. This is a Jump on Negative instruction, where dst is the destination address

Question 2:.....[Marks 5+5+5]

Consider three CPUs, having Main Memory and

a. Only Stack
b. Only Accumulator
c. Accumulator and Registers

Note: Of course, PC, IR, MAR, MBR etc. are available.

Draw the basic block diagrams of these CPUs. Write the instruction set (assume your own design) required for these machines to execute "A=(B+C)-(D*E)/F". Then write assembly language programs for these CPUs for computing "A=(B+C)-(D*E)/F".
[Assume values for B C D E and F are available in memory].

Question 3:.....[Marks 10]

What is the above addressing mode called? Explain with an example of the advantage taking the "round about approach" rather than directly keeping the address of the operand in "Address" field of the instruction.