# Processor Fundamentals Test 1

Monday, 15 November 2021 5:37 PM

Processor FUndamen...



**Q1) (a)** Describe what is meant by Von Neumann model?
It was first microprocessor model(s) describes:
1) Both inst. and data in binary form are indistinguishable and are kept in same memory.
2) It is a single processor, made up of ALU, CU and MU.
3) It makes use input, storage and output devices
4) Its a serial machine.
...[2]

**(b)** Describe the stages of the fetch/decode/execute/reset cycle, explaining how the special registers in the processor are used.

You should use as an example the processing of following instruction:

```
2024   JMP 3000
```

Updating PC

2024 | JMP 3000 — PC → MAR, MDR → CIR; PC: 2024, 2025, 3000; MAR: 2024; MDR: JMP 3000; CIR: JMP 3000

...[2]



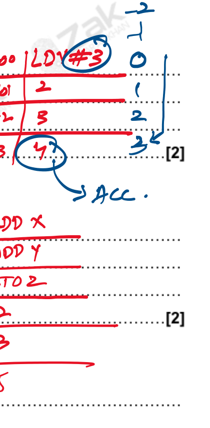


**(c)** A processor will allow the use of a variety of modes of addressing. Explain these terms, using an example in each case. You may wish to illustrate your answer with a diagram.

**(i)** Indirect addressing:
LDI <address>
go to address, pick up add. of data
go to address and bring data from that add.
LDI 300
...[2]

**(ii)** relative addressing:
Displace from curr. inst add #n times and bring data to ACC.
LDY #n
...[2]

**(iii)** symbolic addressing;
Labels




...[2]

**(iv)** Indexed / Immediate / Direct addressing.
...[2]



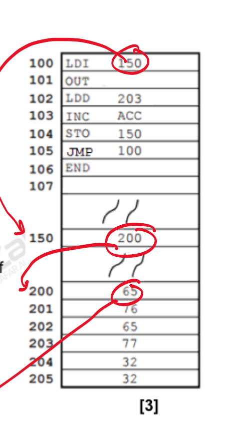


**(d)** The diagram shows a program loaded in main memory starting at location 100.
Locations 200 onwards contain data which are used by the program.

**(i)** The instruction at address 102 is fetched.

ACC: 77
PC: 103

Show the contents of the registers after execution.
Write on the diagram to explain. [2]




**(ii)** The instruction at address 100 is fetched. Show the contents of the registers after execution.

ACC: 65 MAR: 200

Write on the diagram to explain. [3]





OP-CODE

| INSTRUCTION | MACHINE CODE |
|---|---|
| LDI | 00101010 |
| LDM # | 00010110 |

Show the machine code for the following instructions in 16 bits word registers, where 8 bits are used as OP-CODE.
(& denotes a hexadecimal number, e.g. &4A):

**(iii)** LDI &96 (Hex)

| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

[2]

**(iv)** LDM #&FB

| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

[2]

**(v)** A programmer makes the statement:
"For this instruction set, some of the instructions do not require an operand"
What is an operand and what is operand size in above word?
part of the inst which is acted upon by the OP-CODE.
...[2]





**(e)** The following program is to be executed. Shown are:
- the first four instructions only of this program
- the memory locations which are accessed by this program.

| Address | Main Memory |
|---|---|
| 100 | LDR 200 |
| 101 | LDX 200 |
| 102 | LDI 201 |
| 103 | LDX 201 |
| 200 | [illegible] |
| 201 | [illegible] |
| 202 | [illegible] |
| 203 | [illegible] |
| 204 | [illegible] |
| 216 | [illegible] |
| 217 | [illegible] |

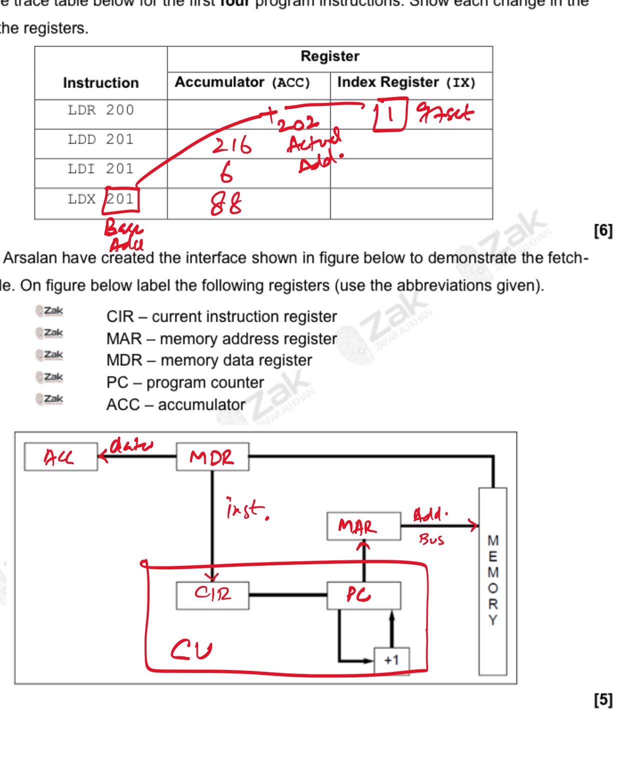




Complete the trace table below for the first **four** program instructions. Show each change in the contents of the registers.

| Instruction | Accumulator (ACC) | Index Register (IX) |
|---|---|---|
| LDR 200 | | |
| LDX 200 | | |
| LDI 201 | | |
| LDX 201 | | |

**Q2) (a)** Ali and Akbar have created the interface shown in figure below to demonstrate the fetch-execute cycle. On figure below label the following registers (use the abbreviations given):
- CIR - current instruction register
- MAR - memory address register
- MDR - memory data register
- PC - program counter
- ACC - accumulator



[5]





**Q3)**

**(a)** Explain the term low-level language.
A language that is directly understood by the microprocessor e.g. assembly language
...[1]

**(b)** Using the assembly language instruction LDD &7A, explain the term opcode.
Part of the inst. code ... in ... assembly language
...[1]





Refer to the table below to answer the rest of the questions

| Instruction | | Explanation |
|---|---|---|
| Op Code | Operand | |
| LDM | #n | Immediate addressing. Load the number n to ACC |
| LDD | <address> | Direct addressing. Load the contents of the location at the given address to ACC |
| LDI | <address> | Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC |
| LDX | <address> | Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC |
| LDR | #n | Immediate addressing. Load the number n to IX |
| STO | <address> | Store the contents of ACC at the given address |
| ADD | <address> | Add the contents of the given address to the ACC |
| INC | <register> | Add 1 to the contents of the register (ACC or IX) |
| DEC | <register> | Subtract 1 from the contents of the register (ACC or IX) |
| JMP | <address> | Jump to the given address |
| CMP | <address> | Compare the contents of ACC with the contents of <address> |
| CMP | #n | Compare the contents of ACC with number n |
| JPE | <address> | Following a compare instruction, jump to <address> if the compare was True |
| JPN | <address> | Following a compare instruction, jump to <address> if the compare was False |
| IN | | Key in a character and store its ASCII value in ACC |
| OUT | | Output to the screen the character whose ASCII value is stored in ACC |
| END | | Return control to the operating system |

All questions will assume there is only one general purpose register available (Accumulator)

\# ACC denotes Accumulator
IX denotes Index Register

\# denotes immediate addressing
B denotes a binary number, e.g. B01001010
& denotes a hexadecimal number, e.g. &4A





**(c)** Here is a block of program code, written in non-assembly language.

```
IF X = 5
   THEN B ← 10
END IF
```

Exit

**Where:** X refers to memory location $(45)_{16}$ in hexadecimal
B refers to memory location $(E4)_{16}$ in hexadecimal

Write a sequence of assembly-language instructions that would perform the same operations as the program code above.

| | |
|---|---|
| 30 | LDD &45 |
| 31 | CMP #5 |
| 32 | JPN 35 |
| 33 | LDM #10 |
| 34 | STO &E4 |
| 35 | End. |

...[4]