Unit 2
Basic Computer Architecture

<u>SAP-1 Architecture</u>

The Simple-As-Possible (SAP)-1 computer is a very basic model of a microprocessor explained by Albert Paul Malvino. The SAP-1 design contains the basic necessities for a functional Microprocessor. Its primary purpose is to develop a basic understanding of how a microprocessor works, interacts with memory and other parts of the system like input and output. The instruction set is very limited and is simple.

SAP (Simple-As-Possible)-1 is the first stage in the evolution toward modern computers.

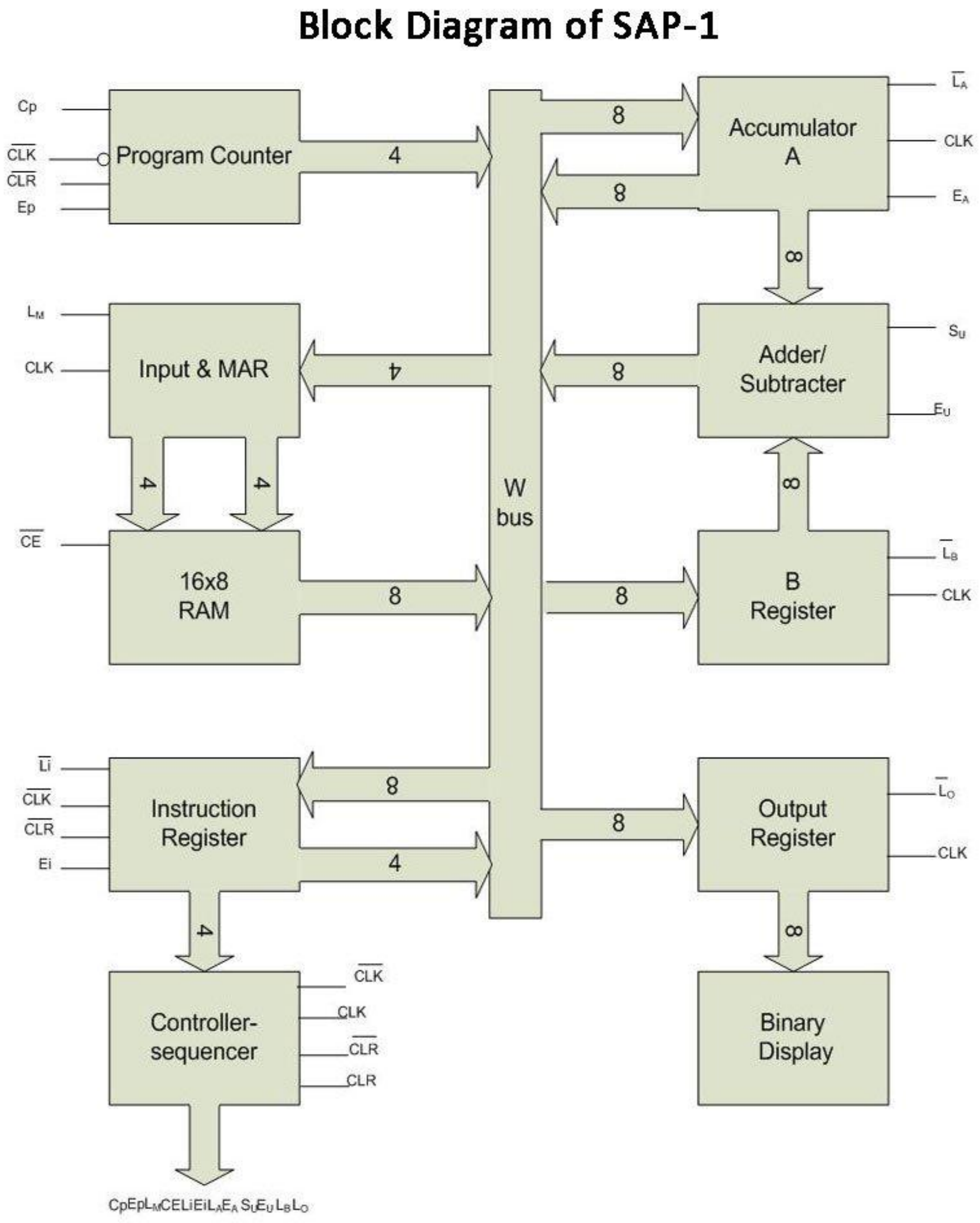


Figure: Architecture of SAP-1 Microprocessor/Computer

SAP is Simple-As-Possible Computer. The type of computer is specially designed for the academic purpose and nothing has to do with the commercial use. The architecture is 8 bits and comprises of 16 X 8 memory i.e. 16 memory location with 8 bits in each location, therefore, need 4 address lines which either comes from the PC (Program Counter which may be called instruction pointer) during computer run phase or may come from the 4 address switches during the program phase. All instructions (5 only) get stored in this memory. It means SAP cannot store program having more than 16 instructions.

SAP can only perform addition and subtraction and no logical operation. These arithmetic operations are performed by an adder/subtractor unit.

There is one general purpose register (B register) used to hold one operand of the arithmetic operation while another is kept by the accumulator register of the SAP-1.

In addition, there are 8 LEDs which work as output unit and connected with the 8 bit output register.
All timely moment of data or activities are performed by the controller/sequencer part of the SAP-1.

Program Counter
- It counts from 0000 to 1111.
- It signals the memory address of next instruction to be fetched and executed.

Inputs and MAR (Memory Address Register)
- During a computer run, the address in PC is latched into Memory Address Register (MAR).

The RAM
- The program code to be executed and data for SAP1 computer is stored here.
- During a computer run, the RAM receives 4-bit addresses from MAR and a read operation is performed. Hence, the instruction or data word stored in RAM is placed on the W bus for use by some other part of the computer.
- It is asynchronous RAM, which means that the output data is available as soon as valid address and control signal are applied.

Instruction Register
- IR contains the instruction (composed of OPCODE+ADDRESS) to be executed by SAP1 computer.

Controller-Sequencer
- It generates the control signals for each block so that actions occur in desired sequence. CLK signal is used to synchronize the overall operation of the SAP1 computer.
- A 12-bit word comes out of the Controller-Sequencer block. This control word determines how the registers will react to the next positive CLK edge.

Accumulator
- It is a 8-bit buffer register that stores intermediate results during a computer run.
- It is always one of the operands of ADD, SUB and OUT instructions.

Adder/Subtractor
- It is a 2's complement adder-subtractor.
- This module is asynchronous (unclocked), which means that its contents can change as soon as the input words change.

B-register
- It is 8-bit buffer register which is primarily used to hold the other operand (one operand is always accumulator) of mathematical operations.

Output Register
- This registers hold the output of OUT instruction.

Binary Display
- It is a row of eight LEDs to show the contents of output register.
- Binary display unit is the output device for the SAP-1 microprocessor.

SAP-1 Instructions

SAP-1 instruction set consists of following instructions

Mnemonic	Operation	OPCODE
LDA	Load addressed memory contents into accumulator	0000
ADD	Add addressed memory contents to accumulator	0001
SUB	Subtract addressed memory contents from accumulator	0010
OUT	Load accumulator data into output register	1110
HLT	Stop processing	1111

The instruction format of SAP-1 Computer is
(XXXX) (XXXX)
The first four bits make the op-code while the last four bits make the operand (address).

Machine cycle and Instruction cycle

SAP1 has six T-states (three fetch and three execute cycles) reserved for each instruction. Not all instructions require all the six T-states for execution. The unused T- state is marked as No Operation (NOP) cycle. Each T-state is called a machine cycle for SAP1. A ring counter is used to generate a T-state at every falling edge of clock pulse. The ring counter output is reset after the 6th T-state.

FETCH CYCLE – T1, T2, T3 machine cycle

EXECUTE CYCLE - T4, T5, T6 machine cycle

- Complete code includes opcode and operand
- One instruction is executed in one instruction cycle
- Instruction cycle may consist of many machine cycles
- For SAP-1, Instruction cycle = Machine cycle
- Instruction cycle = Fetch cycle + Execution cycle
- Fetch cycle is generally same for all instructions
- Complete code includes opcode and operand
- Like LDA 04H $\Longrightarrow$ 0000 0100
- One instruction is executed in one instruction cycle

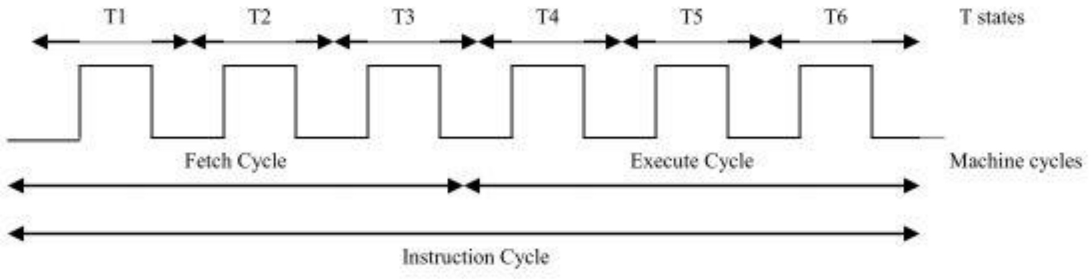


Fig: Instruction Cycle

Fetch and Execution Cycle of SAP-1 instructions

- SAP-1 instruction cycle: 3 clock cycles to fetch and decode phase, 3 clock cycles to execute

- the first three states are:
    1. address
    2. increment
    3. memory
- controller has a 6-bit ring counter which continuously cycles from 000001 up to 100000 then resets (must be set to 000001 when we initialize the computer)
- ring counter is clocked on clock high-to-low transition, most of the other circuits in the computer on clock low-to-high transition

Fetch Cycle

- Address state: enable PC to bus three-state output, MAR load line
- Increment state: enable PC increment (and perhaps wait for memory access time)
- Memory state: enable memory CE, IR load line
- IR is loaded on the low-to-high clock transition, so stabilizes before state 4 is entered

t1: MAR ← PC
t2: PC ← PC +1
t3: IR ← RAM

Execution Cycle -- LDA

- state 4: enable IR to bus three-state output, MAR load line
- state 5: enable memory CE, accumulator load line
- state 6: enable nothing

t4: MAR ← (IR (Address of operand))
t5: Accumulator ← RAM
t6: nothing

Execution Cycle -- ADD

- state 4: enable IR to bus three-state output, MAR load line
- state 5: enable memory CE, register B load line
- state 6: enable add, ALU to bus three-state output, accumulator load line

t4: MAR ← (IR (Address of B))
t5: B ← RAM
t6: Accumulator ← Accumulator + B

Micro program

- each SAP-1 block has some control lines:
    - each three-state driver has an enable line which connects the driver to the bus
    - the program counter also has a count line which, when high, increments the contents on the next low-to-high clock transition
    - all registers have a load line (active low)
    - the ALU has a subtract line which is high for subtraction and low for addition
- implementing the SAP-1 instructions means raising and lowering these control lines at the appropriate times
- these 12 control lines are the micro program word
- controller must, on each clock cycle, produce 12 bits
- some of these bits are on-off (e.g. three-state output lines) and have a "default off" state
- some of these bits are A/B (e.g. the add-subtract line) and have a "default don't care"
- some bits are active high, some are active low

Controller Implementation

- How to generate micro words?
- if a bit is only on during one cycle, connect it to the corresponding ring counter bit
- if a bit is only on for an instruction, connect it to that instruction (as decoded by a 4-bit 1-of-16 decoder)

- if a bit is only on for an instruction and a cycle, connect it to the AND of the ring counter bit and the decoder output
- if a bit is on for multiple instruction and/or cycles, work out the truth table and use AND/OR or multiplexers to implement it
- alternative: use a ROM

SAP-2 Architecture

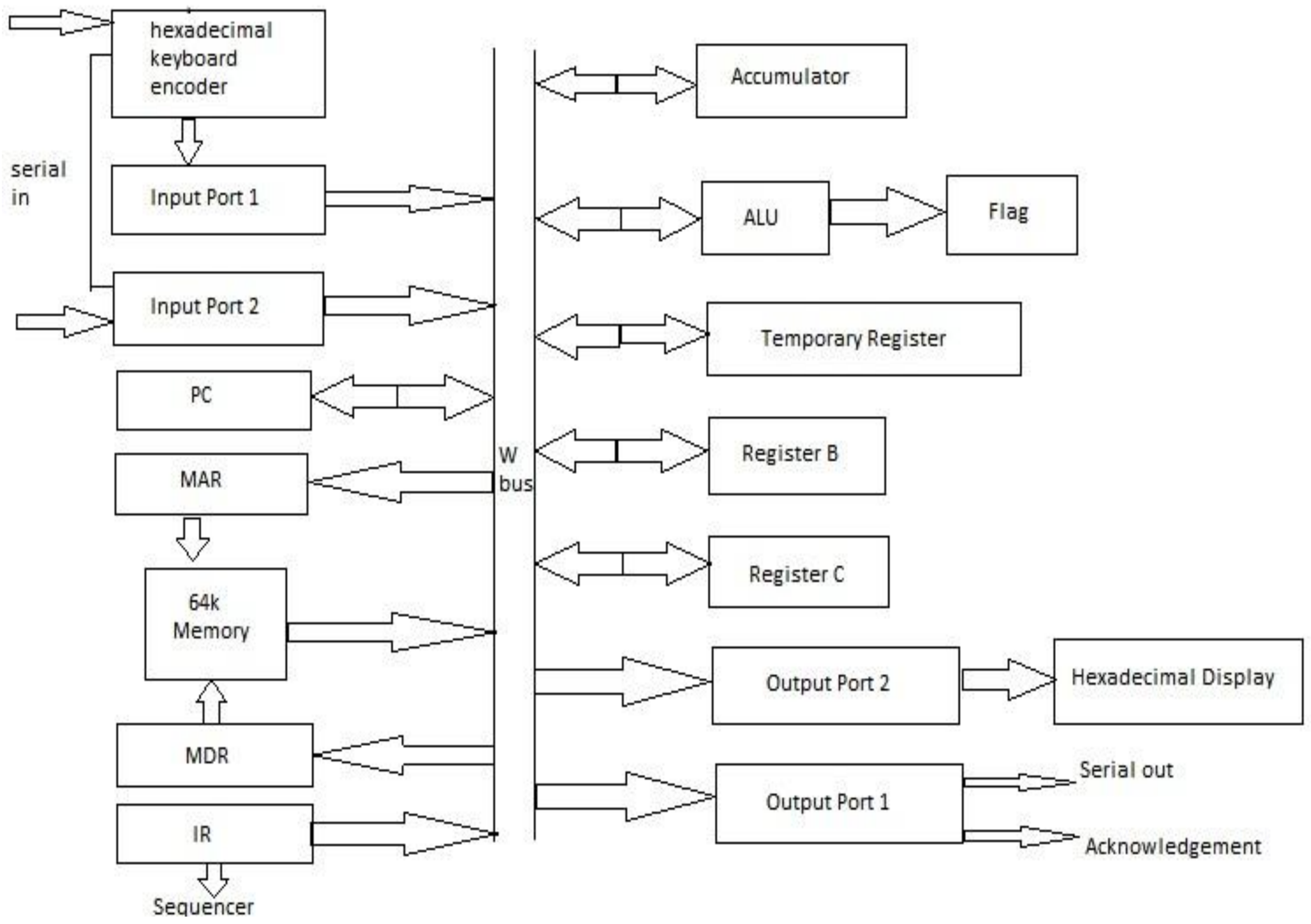


Fig: Architecture of SAP-1 Microprocessor/Computer

- **Hexadecimal Keyboard Encoder**: The hexadecimal keyboard encoder receives the data from outer environment and converts it into hexadecimal form. The system can understand and send them to the input port.
- **Input Ports**: The SAP-2 contains two input ports which input the data in the system in the most convenient way.
- **PC**: PC is the program counter that holds the address of the next instruction to be fetched. It initializes from 0000H to 1111H during the execution.
- **MAR**: MAR is the memory address register that stores the complete format of the address sent by the program counter. It stores the final address of the memory word that needs some computations.
- **64 K Memory**: It contains 64 K memory where data and instruction reside. All the computations are performed relative to the memory.
- **MDR**: MDR is the Memory Data Register which stores the data or operand that is fetched from the memory which is needed for computation.

- **IR**: The IR is the Instruction Register that holds the complete format of the Instruction that is to be executed.
- **Control Sequencer**: It provides necessary timing signals like T0, T1, T2, ….. and control signals providing the direction for executing the program.
- **Accumulator**: The result of all the mathematical operations is stored in accumulator. It is one of the operand of ADD, OUT, SUB instruction. It is also known as processor register.
- **ALU and Flag**: The ALU perform all the arithmetic and logical calculations. The flag reflect the intermediate changes on the values during execution.
- **Temporary register, B, C**: They are the second operand of the mathematical operations. The register B and C is accessible to the programmer.
- **Output Ports**: It consists of two output ports to show the result of OUT instruction.
- **Hexadecimal Display**: Unlike Sap-1 which has binary display, Sap-2 has a hexadecimal display to show outputs in the LEDs.

Architectural differences with SAP-1

- Jump Instructions: loadable PC
- 16-bit program counter
- 8-bit op-code, 42 instructions
- 2 input ports, 2 output ports
- 2K ROM, up to 62K RAM (16-bit addresses) with read and write
- memory data register (MDR) buffers reads and writes
- accumulator can write to bus
- Temporary, B and C registers
- 16 arithmetic and logic operations in ALU
- sign and zero flag

Bidirectional registers

- connect the inputs to the outputs
- load and enable never simultaneously active
- on load, outputs are 3-state, input is taken from the bus
- on enable, inputs are ignored, output goes to the bus
- 1/2 as many pins
- 1/2 as much bus capacitance

Flags

- 2 flip flops, sign flag and zero flag
- set during arithmetic and logic operations to reflect final accumulator contents
- JM jumps only if the sign flag is set (minus result)
- JZ jumps only if the zero flag is set (zero result)
- JNZ jumps if the zero flag is clear (non-zero result)

    Q. What is the value of the sign bit if the accumulator contents are zero?

In-Class Exercise

- work in groups of up to 3
- design a circuit to implement the flags
- inputs are: 8 bits from the accumulator, clock (use the positive-going edge), 1 L_F control line (active high)
- outputs are the flags: Z_F, S_F

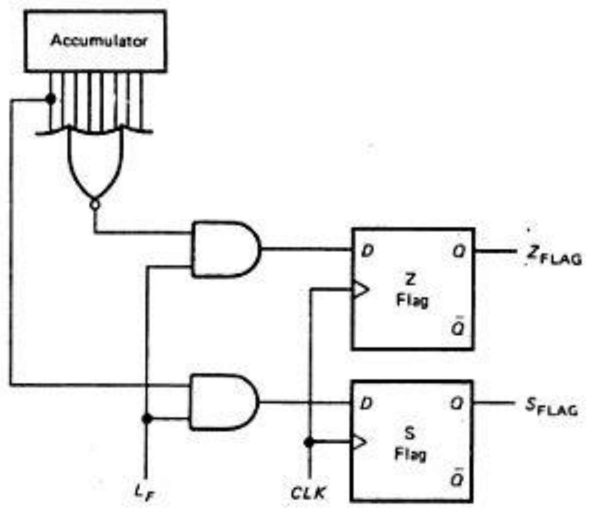


Fig: Setting the flags

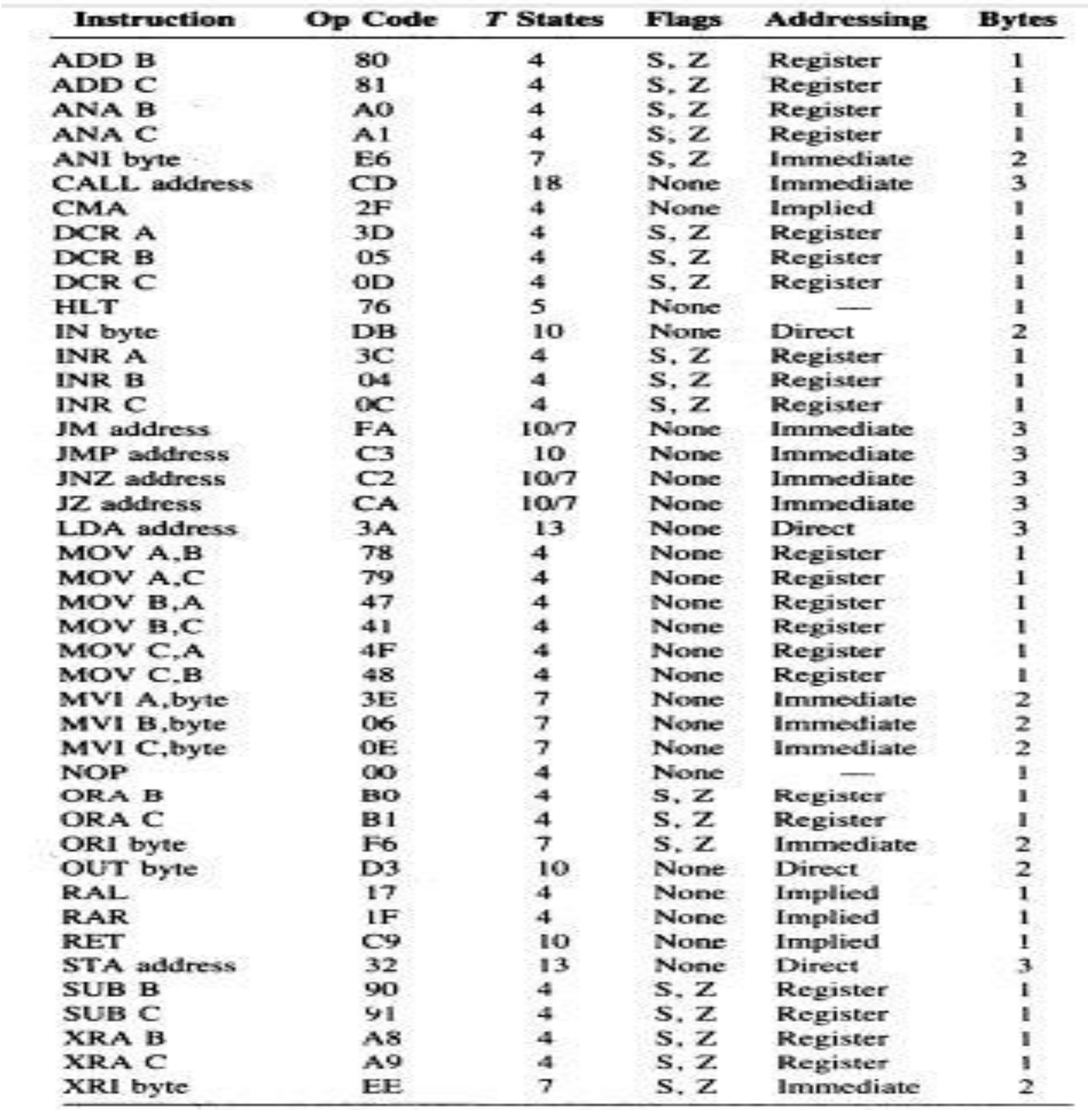
SAP-2 Instruction Sets

- same fetch cycle (T_1, T_2, T_3) as SAP-1
- memory reference instructions: LDA, STA (3 bytes, lower byte before higher byte)
- immediate instructions: MVI reg, value (2 bytes)
- register instructions: MOV, ADD and SUB, INR and DCR, ANA, ORA, XRA, CMA (1 byte), ANI, ORI, XRI (2 bytes)
- jump and call instructions: JMP, JM, JZ, JNZ, CALL (3 bytes), RET (1 byte)
- CALL saves return address in memory FFFEH and FFFFH
- NOP, HLT, RAL, RAR (1 byte), IN, OUT (2 bytes)

In-Class Exercise 1

- hand-assemble the following program starting at address 2000H
- opcodes are: “IN” is DBH, "MOV B,A" is 47H, "DCR A" is 3DH, "MOV C,A" is 4FH, "ANA B" is A0H, "MOV A,C" is 79H, JNZ is 79H
- What does the program do?

```
START   IN 01H
LOOP    MOV B, A
        DCR A
        MOV C, A
        ANA B
        MOV A, C
        JNZ LOOP
DONE
```



Instruction	Op Code	*T* States	Flags	Addressing	Bytes
ADD B	80	4	S, Z	Register	1
ADD C	81	4	S, Z	Register	1
ANA B	A0	4	S, Z	Register	1
ANA C	A1	4	S, Z	Register	1
ANI byte	E6	7	S, Z	Immediate	2
CALL address	CD	18	None	Immediate	3
CMA	2F	4	None	Implied	1
DCR A	3D	4	S, Z	Register	1
DCR B	05	4	S, Z	Register	1
DCR C	0D	4	S, Z	Register	1
HLT	76	5	None	—	1
IN byte	DB	10	None	Direct	2
INR A	3C	4	S, Z	Register	1
INR B	04	4	S, Z	Register	1
INR C	0C	4	S, Z	Register	1
JM address	FA	10/7	None	Immediate	3
JMP address	C3	10	None	Immediate	3
JNZ address	C2	10/7	None	Immediate	3
JZ address	CA	10/7	None	Immediate	3
LDA address	3A	13	None	Direct	3
MOV A,B	78	4	None	Register	1
MOV A,C	79	4	None	Register	1
MOV B,A	47	4	None	Register	1
MOV B,C	41	4	None	Register	1
MOV C,A	4F	4	None	Register	1
MOV C,B	48	4	None	Register	1
MVI A,byte	3E	7	None	Immediate	2
MVI B,byte	06	7	None	Immediate	2
MVI C,byte	0E	7	None	Immediate	2
NOP	00	4	None	—	1
ORA B	B0	4	S, Z	Register	1
ORA C	B1	4	S, Z	Register	1
ORI byte	F6	7	S, Z	Immediate	2
OUT byte	D3	10	None	Direct	2
RAL	17	4	None	Implied	1
RAR	1F	4	None	Implied	1
RET	C9	10	None	Implied	1
STA address	32	13	None	Direct	3
SUB B	90	4	S, Z	Register	1
SUB C	91	4	S, Z	Register	1
XRA B	A8	4	S, Z	Register	1
XRA C	A9	4	S, Z	Register	1
XRI byte	EE	7	S, Z	Immediate	2

Fig: SAP-2 Instruction Sets

(I have explained in class.)

Differences between SAP-1 and SAP-2 Architecture

SAP-1	SAP-2
It has 8-bit bus.	It has 16-bit bus.
PC is 4-bit.	PC is 16-bit.

It does not have hexadecimal keyboard encoder.	It has hexadecimal keyboard encoder.
It has single input.	It has two input ports.
MAR receives 4-bit address from PC.	MAR receives 16-bit address from PC.
It does not have ROM.	It has 2 KB ROM.
It has 16 Byte memory.	It has 62 KB memory.
It does not have MDR.	It has MDR.
It has only adder/subtractor.	It has ALU.
It does not have flag.	It has 2 flags.
It does not have temporary register.	It has temporary register.
It has single register (B).	It has 2 registers (B and C).
It has single output port.	It has 2 output ports.
It has 5 instruction sets.	It has 42 instruction sets.