HLO: High level operation

MO: Micro-operation

Common micro-operations for all instructions:

Fetch cycle:

1st cycle: load PC value to MAR

2nd cycle: RAM to MDR and PC=PC +1

3rd cycle: MDR to IR

Micro operations for individual instructions (execution cycle):

1. LDA address

Opcode: 07H

HLO: ACC←Memory[Address]

MO: i) MAR ← address (operand)

ii) ACC ← MDR

4th cycle: load PC value to MAR

5th cycle: load RAM[MAR] value to MDR and PC++

6th cycle: load MDR to MAR

7th cycle: load RAM[MAR] value to MDR

8th cycle: load MDR data to Accumulator

2. STA address

Opcode:

HLO: Memory[Address] ← ACC

MO:

4th cycle: load PC to MAR

5th cycle: load RAM[MAR] to MDR, PC++ //MDR holds the operand

6th cycle: load MDR to MAR

7th cycle: load ACC to MDR

8th cycle: Write MDR data to RAM

3. MOV Acc, B

Opcode: 03H

HLO: ACC ← B

MO:

4th cycle: load data from B to Accumulator

4. MOV B, Acc

Opcode: 04H

HLO: B← ACC

MO:

4th cycle: load Accumulator to B

5. MOV Acc, immediate

Opcode: 05H

HLO: ACC← immediate

MO:

4th cycle: load PC value to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: load MDR to Accumulator

6. IN

Opcode: 01H

HLO: ACC← Input

MO:

4th cycle: load Input to Accumulator

7. OUT

Opcode: 02H

HLO: Output← ACC

MO:

4th cycle: Load Accumulator to Output

8. ADD B

Opcode: 08H

HLO: ACC ← ACC+B

MO:

4th cycle: Load register B to TEMP\_REGISTER

5th cycle: Select ALU Operation for ADD

6th cycle: Load ALU output to Accumulator

9. ADC B

Opcode:

HLO: ACC← ACC+ B +C (Carry)

MO:

4th cycle: Load register B to TEMP\_REGISTER

5th cycle: Select ALU Operation for ADC

6th cycle: Load ALU output to Accumulator

10. SUB B

Opcode:

HLO: ACC← ACC- B

MO:

4th cycle: load B to TEMP\_REGISTER

5th cycle: select ALU operation for SUB

6th cycle: Load ALU output to Accumulator.

11. SBB B

Opcode:

HLO: ACC← ACC - B -Bo(Borrow)

MO:

4th cycle: load B to TEMP\_REGISTER

5th cycle: select ALU operation for SBB

6th cycle: Load ALU output to Accumulator.

12. ADD immediate

Opcode:

HLO: ACC← ACC +immediate

MO:

4th cycle: Load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: Load MDR to TEMP\_REGISTER

7th cycle: Select ALU operation for ADD

8th cycle: Load ALU output to Accumulator.

13. SUB [address]

Opcode:

HLO: ACC← ACC – RAM[address]

MO:

4th cycle: Load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: Load MDR to MAR

7th cycle: load RAM[MAR] to MDR

8th cycle: load MDR to TEMP\_REGISTER

9th cycle: Select ALU operation for SUB

10th cycle: Load ALU output to Accumulator.

14. CMP B

Opcode:

HLO: ACC not change, but flags will be changed according to (ACC-B)

MO:

4th cycle: load B to TEMP\_REGISTER

5th cycle: Select ALU operation for SUB (no load is needed to Accumulator)

15. TEST B

Opcode:

HLO: ACC not change, but flags will be changed according to (ACC & B)

MO:

4th cycle: load B to TEMP\_REGISTER

5th cycle: Select ALU operation for AND (no load is needed to Accumulator)

16. JC address

Opcode:

HLO: Jump to address when carry flag is 1.

MO: If carry is 1,

4th cycle: load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: load MDR to PC

17. JE address

Opcode:

HLO: Jump to address when zero flag is 1.

MO: If zero flag is 1,

4th cycle: load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: load MDR to PC

18. PUSH

Opcode:

HLO: STACK← ACC

MO:

4th cycle: Decrement SP and load accumulator to MDR.

5th cycle: Load SP to MAR

6th cycle: Write MDR to RAM[MAR]

19. POP

Opcode:

HLO: ACC← STACK

MO:

4th cycle: load SP to MAR

5th cycle: load RAM[MAR] to MDR

6th cycle: increment SP and load MDR to Accumulator

20. CALL address

Opcode:

HLO: Calls a subroutine (at the specified address) unconditionally.

i) Save next instruction address to stack

ii) Jump to the operand address

MO:

4th cycle: Load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: Load MDR to TEMP\_REGISTER and decrement SP

7th cycle: Load SP to MAR

8th cycle: Load PC to MDR

9th cycle: Write MDR to RAM[MAR]

10th cycle: Load TEMP\_REGISTER to PC

21. RET

Opcode:

HLO: Returns from current subroutine to the caller unconditionally.

MO:

4th cycle: Load SP to MAR

5th cycle: load RAM[MAR] to MDR

6th cycle: Load MDR to PC and Increment SP

22. JMP address

Opcode:

HLO: Jumps unconditionally to address.

MO:

4th cycle: Load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: Load MDR to PC

23. HLT

Opcode: 00H

HLO: Halts execution.

MO: 4th cycle: activate halt pin to stop the clock.

24. NOP

Opcode: 06H

MO:

4th cycle: Nothing to do, send back to the start of fetch cycle.

25. SHL

Opcode:

HLO: ACC ← ACC<<1

MO:

4th cycle: Select shift left mode of Accumulator

26. SHR

Opcode:

HLO: ACC ← ACC>>1

MO:

4th cycle: Select shift right mode of Accumulator

27. AND B

Opcode:

HLO: ACC← ACC & B

MO:

4th cycle: Load B to TEMP\_REGISTER

5th cycle: SELECT ALU operation AND

6th cycle: Load ALU output to Accumulator

28. XOR immediate

Opcode:

HLO: ACC← ACC ⊕ immediate

MO:

4th cycle: Load PC to MAR

5th cycle: load RAM[MAR] to MDR and PC++

6th cycle: Load MDR to TEMP\_REGISTER

7th cycle: Select ALU operation for XOR

8th cycle: Load ALU output to Accumulator.

**Cycle Description:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Macro-instruction | Op-code | Description | Total  T-States | TSt  at  e | Micro-Operation | Active | CON | | | | |
| ALL | - | Fetch | 4 | T1 | MAR ←PC | LM\_BAR, EP\_BAR |  |  |  |  |  | |
| T2 | PC←PC+1,  MDR ←  RAM[MAR] | CP, LDR\_BAR,  R\_BAR, EM\_BAR |  |  |  |  |  | |
| T3 | IR ←MDR | EDB\_BAR,  LIR\_BAR |  |  |  |  |  | |
| T4 | LOAD from IR | LOAD |  |  |  |  |  | |
| HALT | 00H | Halts  execution | 5 | T5 | HALT | HLT |  |  |  |  |  | |
| IN | 01H | ACC ←  input\_port | 5 | T5 | ACC ←  input\_port | LA\_BAR, EIN\_BAR,  RESET |  |  |  |  |  | |
| OUT | 02H | output\_port  ←ACC | 5 | T5 | output\_port  ←ACC | LO\_BAR, EA\_BAR,  RESET |  |  |  |  |  | |
| MOV B,  ACC | 03H | B ←ACC | 5 | T5 | B ←ACC | LB\_BAR, EA\_BAR,  RESET |  |  |  |  |  | |
| MOV ACC,  B | 04H | ACC ←B | 5 | T5 | ACC ←B | LA\_BAR, EB\_BAR,  RESET |  |  |  |  |  | |
| NOP | 06H | No Operation | 5 | T5 | NOP | RESET |  |  |  |  |  | |
| LDA  address | 07H | ACC ←  RAM[address] | 9 | T5 | MAR ←PC | LM\_BAR, EP\_BAR |  |  |  |  |  | |
| T6 | PC←PC+1,  MDR ←  RAM[MAR] | CP, LDR\_BAR,  R\_BAR, EM\_BAR |  |  |  |  |  | |
| T7 | MAR ←MDR | LM\_BAR,  EDB\_BAR |  |  |  |  |  | |
| T8 | MDR ←  RAM[MAR] | LDR\_BAR, R\_BAR,  EM\_BAR |  |  |  |  |  | |
| T9 | ACC ←MDR | LA\_BAR,  EDB\_BAR, RESET |  |  |  |  |  | |
| MOV ACC, immediate | 05H | ACC ←immediate | 7 | T5 | MAR ←PC | LM\_BAR, EP\_BAR,  EM\_BAR |  |  |  |  |  | |
| T6 | PC←PC+1,  MDR ←  RAM[MAR] | CP, LDR\_BAR,  R\_BAR, EM\_BAR |  |  |  |  |  | |
| T7 | ACC ←MDR | LA\_BAR,  EDB\_BAR, RESET |  |  |  |  |  | |
| ADD B | 08H | ACC ←ACC +  B | 7 | T5 | TEMP ←B |  |  |  |  |  |  | |
| T6 | ACC ←ACC + TEMP |  |  |  |  |  |  | |
| T7 |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  | | | | |