CPU Architecture Documentation

Made by: sherbodev

Date: January 17, 2026

Credits: Inspired by Sebastian Lague's Digital Logic Simulator

Contents

1 Structure

1.1 Registers

Register	**Size (bits)**	**Name**
PC	8	Program Counter
CIR_Opcode	4	Current Instruction Register
CIR_Operand	8	Current Instruction Register
MAR	8	Memory Address Register
MDR	8	Memory Data Register
ACC	8	Accumulator
FLAG	4	G, E
RAM_Opcode	256×4	Random Access Memory
RAM_Operand	256×8	Random Access Memory

RAM is split into: ***224×12 ROM + 32×12 RAM***

1.2 Operations

Mnemonic	**Opcode**	**Details**
NOP	0	No Operation
LDA	1	Load RAM $\rightarrow$ ACC
LDI	2	Load Immediate $\rightarrow$ ACC
STA	3	Store ACC $\rightarrow$ RAM
JMP	4	Jump Unconditionally
JEQ	5	Jump if E FLAG
JGT	6	Jump if G FLAG
CMP	7	Set FLAG
ADD	8	Add ACC += RAM
ADI	9	Add ACC += Immediate
SUB	10	Sub ACC -= RAM
SBI	11	Sub ACC -= Immediate
SHF	12	Shift ACC $\ll$= RAM
AND	13	And ACC &= RAM
ORR	14	Or ACC |= RAM
XOR	15	Xor ACC $\oplus =$ RAM

2 Fetch

```
MAR <- PC
CIR_Opcode <- RAM[MAR]_Opcode
CIR_Operand <- RAM[MAR]_Operand
PC <- PC + 1
```

3 Decode

Opcode -> 4 bits
Operand -> 8 bits (Data/Address)

Special Cases

`0` -> NOP
`3` -> STA

```
MAR <- CIR_Operand
```

`4,5,6` -> jumps

```
MDR <- CIR_Operand
```

ACC/FLAGS <- x

`1,7,8,10,12,13,14,15` (fetch address)

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
```

`2,9,11` (Immediate)

```
MDR <- CIR_Operand
```

4 Execute

0 – NOP

```
-- No Operation --
```

1 – LDA

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- MDR
```

2 – LDI

```
MDR <- CIR_Operand
ACC <- MDR
```

3 – STA

```
MAR <- CIR_Operand
RAM[MAR]_Operand <- ACC
```

4 – JMP

```
MDR <- CIR_Operand
PC <- MDR
```

5 – JEQ

```
MDR <- CIR_Operand
if E:
    PC <- MDR
```

6 – JGT

```
MDR <- CIR_Operand
if G:
    PC <- MDR
```

7 – CMP

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
FLAG <- compare(ACC, MDR)
```

8 – ADD

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC + MDR
```

9 – ADI

```
MDR <- CIR_Operand
ACC <- ACC + MDR
```

10 – SUB

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC - MDR
```

11 – SBI

```
MDR <- CIR_Operand
ACC <- ACC - MDR
```

12 – SHF

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC << MDR
```

13 – AND

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC & MDR
```

14 – ORR

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC | MDR
```

15 – XOR

```
MAR <- CIR_Operand
MDR <- RAM[MAR]_Operand
ACC <- ACC ^ MDR
```

5 Connections

PC

```
ALU 1-> | PC | *-> MAR
```

CIR

```
RAM 1-> | CIR_Operand | /2-> MAR
        |             | \3-> MDR
```

MAR

```
PC 1->\ | MAR | *-> RAM[]
CIR 2->/
```

MDR

```
CIR 1->\ | MDR | 3-> ALU
RAM 2->/
```

ACC

```
ALU 1-> | ACC | 2-> MDR
```

FLAG

```
ALU 1*-> | FLAG | 2*-> ALU
```

RAM

```
ALU 1-> | RAM | /2-> CIR
MAR *-> |     | \3-> MDR
```

6 Cycle

```
10 tick cycle:
4 - FETCH
4/2 - DECODE
2 - EXECUTE
```

7 Buses

```
store MAR
pulse MAR

store MDR
pulse MDR

pulse EXECUTE
```