

**Computer Architecture Project**

**SEM-Team 10**



**Names**

* **Moamen Hassan**
* **Hussein Youssef**
* **Hassan Osama**
* **Mohamed Magdy**

# **Design**

The Processor is similar to PDP11-based microprocessor that can execute the program loaded in its ram.

The Processer has 3 busses to write data on them. The first Two ‘A’ and ‘B’ can read from registers and The Third ‘C’ can write data into register.

We assume that we have two temp registers ‘X’ and ‘Y’.

**Components**

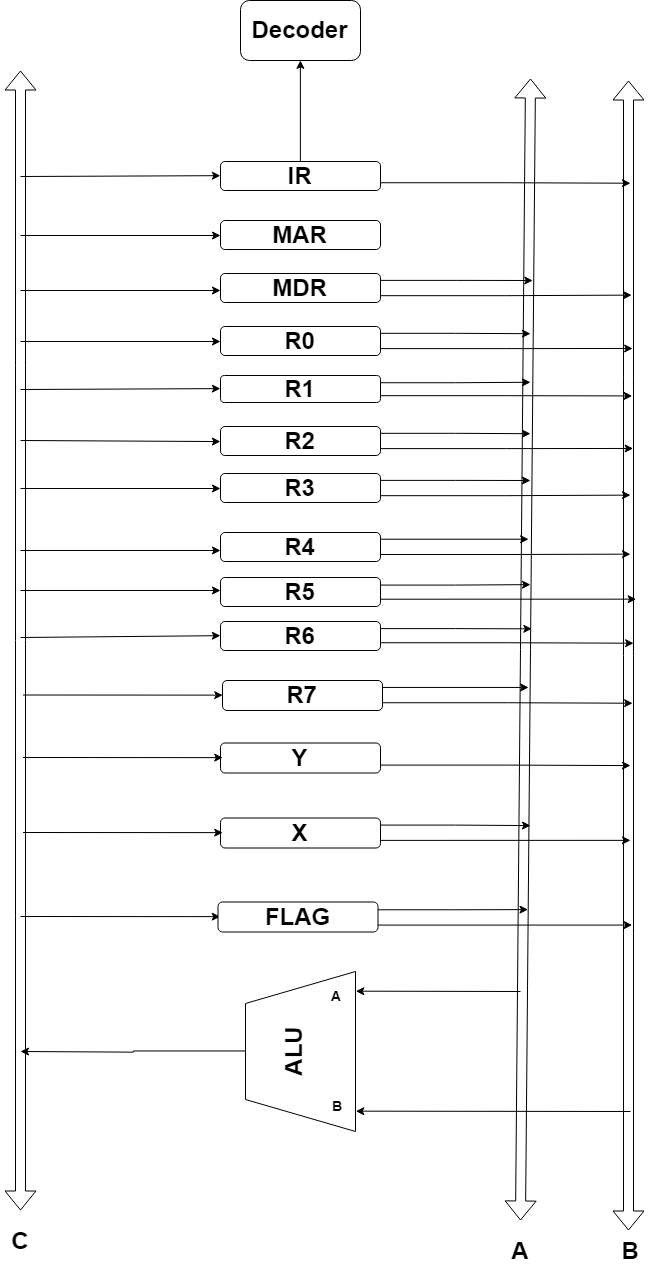
Eight Registers numbered from **R0** to **R7**.

Two temp registers **X**, **Y**.

Four special purpose registers **IR**, **MAR**, **MDR**, **FLAG**.

* **IR**: Instruction Register
* **MAR**: Memory Address Register
* **MDR**: Memory Data Register
* **FLAG**: Status Flag Register contains (**C**, **Z**, **N**, **P**, **O**)
* **C**: Carry Flag.
* **Z**: Zero Flag (1 if ALU result is 0).
* **N**: Negative Flag (1 if ALU result sign is Negative).
* **P**: Parity Flag (1 if ALU result is even).
* **O**: Overflow Flag (1 if P+P=N or N+N=P or P-N=N or N-P=P)

**ALU** that makes operations such as add , sub , addc, subc.



**Mapping the instructions.**

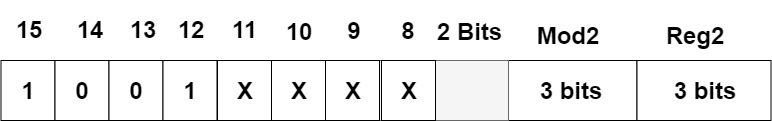
**Two Operand Instructions.**

The selector of the instructions is 4 bits (most 4 significant bits).



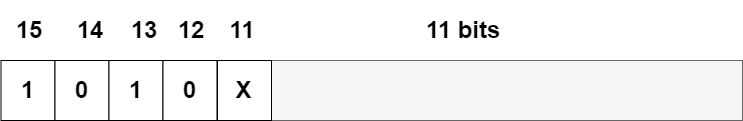
|  |  |
| --- | --- |
| opcode | Two Operand Instructions |
| 0000 | **MOV** |
| 0001 | **ADD** |
| 0010 | **ADC (Add with Carry)** |
| 0011 | **SUB** |
| 0100 | **SBC (Sub with Carry)** |
| 0101 | **AND** |
| 0110 | **OR** |
| 0111 | **XNOR** |
| 1000 | **CMP (Compare)** |

**One Operand Instructions**

****

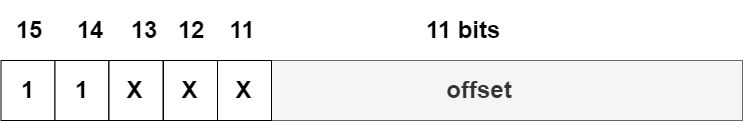
|  |  |
| --- | --- |
| opcode | Two Operand Instructions |
| 0000 | **INC (Increment)** |
| 0001 | **DEC (Decrement)** |
| 0010 | **CLR (Clear)** |
| 0011 | **INV (Inverter)** |
| 0100 | **LSR (Logic Shift Right)** |
| 0101 | **ROR (Rotate Right)** |
| 0110 | **RRC (Rotate Right with Carry)** |
| 0111 | **ASR (Arithmetic Shift Right)** |
| 1000 | **LSL (Logic Shift Left)** |
| 1001 | **ROL (Rotate Left)** |
| 1010 | **RLC (Rotate Left with Carry)** |

**No Operand Instructions**

****

|  |  |
| --- | --- |
| opcode | Two Operand Instructions |
| 0 | **HLT (Halt)** |
| 1 | **NOP (No Operation)** |

**Branches Instructions**

****

|  |  |
| --- | --- |
| opcode | Two Operand Instructions |
| 000 | **BR (Branch unconditionally)** |
| 001 | **BEQ (Branch if equal)** |
| 010 | **BNE (Branch if not equal)** |
| 011 | **BLO (Branch if Lower)** |
| 100 | **BLS (Branch if Lower or same)** |
| 101 | **BHI (Branch if Higher)** |
| 110 | **BHS (Branch if Higher or same)** |

**Bonus Instructions**

**1 – JSR Address**

****

**2 – RTS**

****

**3 – IRET**

****

**4 – INTERRUPT**

****

**Addressing Modes**

|  |  |
| --- | --- |
| opcode | Two Operand Instructions |
| 000 | **Register Mode** |
| 001 | **Auto-Increment** |
| 010 | **Auto-Decrement** |
| 011 | **Indexed** |
| 100 | **Register Mode Indirect** |
| 101 | **Auto-Increment Indirect** |
| 110 | **Auto-Decrement Indirect** |
| 111 | **Indexed Indirect** |

**Analyzing the Design :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Addressing Mode** | **Instruction** | **Clock Cycles** | **MA** |
| **Register Mode** | **MOV Ro,R1** | **4** | **1** |
| **ADD R3,R5** | **4** | **1** |
| **Auto Increment Mode** | **ADD Ro,(R1)+** | **6** | **3** |
| **AND (R1)+,Ro** | **6** | **2** |
| **Auto Decrement Mode** | **XNOR Ro,-(R1)** | **6** | **3** |
| **OR (R1)+,-(Ro)** | **8** | **4** |
| **Indexed** | **Add X(Ro),R1** | **5** | **3** |
| **SUB X(Ro),(R1)+** | **8** | **5** |
| **MOV X(R0),-(R1)** | **8** | **5** |
| **Register Indirect** | **MOV @(Ro),@(R1)** | **7** | **3** |
|  | **ADD @(Ro),(R1)+** | **8** | **4** |
|  | **Add @(R1),-(Ro)** | **8** | **4** |
|  | **Add @(R1),X(Ro)** | **7** | **5** |
|  | **Add @(Ro),R1** | **5** | **2** |
| **Auto Increment Indirect** | **MOV @(Ro)+,R1** | **6** | **3** |
| **MOV @(Ro)+,(R1)+** | **9** | **4** |
| **ADD @(Ro)+,-(R1)** | **9** | **5** |
| **SUB @(Ro)+,X(R1)** | **8** | **6** |
| **XOR @(Ro)+,@R1** | **8** | **5** |
| **Auto Decrement Indirect** | **CLR @-(Ro)** | **7** | **3** |
|  | **MOV @-(Ro),(R1)+** | **9** | **4** |
|  | **ADD @-(Ro),-(R1)** | **9** | **5** |
| **Indexed Indirect** | **Add @X(Ro), R1** | **9** | **4** |

**Add X(R0) , @R1 (5 MA)**

1. **Instruction Fetch/Decode (3 Cycles)**
2. **PCout A , Transfer A , MARin , Read**
3. **PCout A , F = A+1 , PCin , WMFC**
4. **MDRout A , Transfer A , IRin**
5. **First Operand Fetch (4 Cycles)**
6. **PCout A , Transfer A , MARin , Read**
7. **PCout A , F = A+1 , PCin , WMFC**
8. **MDRout A , R0out B , F = A+B , MARin , Read , WMFC**
9. **MDRout A , Transfer A , Xin**
10. **Second Operand Fetch (1 Cycle)**
11. **R1out A , Transfer A , MARin , Read , WMFC**
12. **Operation and Save (1 Cycle)**
13. **MDRout A , Xout B , F = A+B , MDRin , Write**

**BRQ label (1 MA)**

1. **Instruction Fetch/Decode (3 Cycles)**
2. **PCout A , Transfer A , MARin , Read**
3. **PCout A , F = A+1 , PCin , WMFC**
4. **MDRout A , Transfer A , IRin**
5. **Operation and Save (1 Cycle)**
6. **Address of IRout A , PCout B , F = A+B , PCin (if Z(bar) then END)**

**HALT (1 MA)**

1. **Instruction Fetch/Decode (3 Cycles)**
2. **PCout A , Transfer A , MARin , Read**
3. **PCout A , F = A+1 , PCin , WMFC**
4. **MDRout A , Transfer A , IRin**

**JSR R0 , Routine (3 MA)**

1. **Instruction Fetch/Decode (3 Cycles)**
2. **PCout A , Transfer A , MARin , Read**
3. **PCout A , F = A+1 , PCin , WMFC**
4. **MDRout A , Transfer A , IRin**
5. **Operand (Address) Fetch (3 Cycles)**
6. **PCout A , Transfer A , MARin , Read**
7. **PCout A , F = A+1 , PCin , WMFC**
8. **MDRout A , Transfer A , Xin**
9. **Operation and Save (3 Cycles)**
10. **R6(SP)out A , F = A - 1 , R6in , MARin**
11. **PCout A , Transfer A , MDRin , Write**
12. **Xout A , transfer A , PCin , WMFC**

**CPI = 25 / 4 = 6 Cycles per instruction on Average**