COMPUTER ARCHITECTURE AND ORGANISATION

**CECSC07**

****

Group project on

**CPU Emulator**

**with Assembler**

Shivam Sharma (2019UCO1573)

Vishal Verma (2019UCO1563)

Rijul Garg (2019UCO1570)

Suraj Kumar (2019UCO1559)

Riju Tejas Singh (2019UCO1572)

Shrey Rastogi (2019UCO1562)

COE, Sec. 1

Semester 3 ( 2020-2021)

This CPU consists of a RAM of size 1024 \* 16 bits i.e. **2KB**. There are separate buses for data and address, data bus is 16 bits wide and address bus is 10 bits wide. There 15th bit of instruction is the **mode bit** which takes the values

0 : Direct Mode

1 : Indirect Mode

Bits 14th to 10th are for the opcode which is 5 bits wide and bits 9th to 0th are for the address which is 10 bits wide.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| I | OPCODE | ADDRESS |

*Instruction Format*

The following is the Instruction Set used for the assembler as well as the CPU, it consists of 24 instructions.

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Instruction Code** | **Operation** |
| LDAC | 00001 | AC = M[AR] |
| STAC | 00010 | M[AR] = AC |
| ADD | 00011 | AC = AC + DR  If (AC == 0) Z=1  Else Z=0 |
| SUB | 00100 | AC = AC - DR  If (AC == 0) Z=1  Else Z=0 |
| AND | 00101 | AC = AC ^ DR  If (AC ==0 ) Z=1  else Z=0 |
| OR | 00110 | AC = AC v DR  If (AC == 0) Z=1  else Z=0 |
| XOR | 00111 | AC = AC **⊕** DR  If (AC ==0) Z=1  else Z=0 |
| JC | 01000 | If C ==1  AR = M[AR] |
| JNC | 01001 | If C==0  AR = M[AR] |
| JMP | 01010 | AR = M[AR] |
| JZ | 01011 | If Z==1  AR = M[AR] |
| JNZ | 01100 | If Z==0  AR = M[AR] |
| INAC | 01101 | AC = AC + 1  If (AC == 0) Z=1  else Z=0 |
| DECAC | 01101 | AC = AC - 1  If (AC == 0) Z=1  else Z=0 |
| NOT | 01111 | AC = AC **⊕** 1  If (AC == 0) Z=1  else Z=0 |
| CLAC | 10000 | AC =0  Z=1 |
| SHL | 10001 | shift left AC |
| SHR | 10010 | shift right AC |
| CIL | 10011 | circular shift left AC |
| CIR | 10100 | circular shift right AC |
| ASL | 10101 | Arithmetic shift left AC |
| ASR | 10110 | Arithmetic shift right AC |
| HLT | 10111 | Processor halted |
| NOP | 11000 | No operation |

This CPU contains 5 processor registers namely PC (program counter), AR (address register), AC (accumulator), DR (data register) and IR (instruction register).

The status section consists of 3 flags which are ZF (zero flag), CF (carry flag) and MF (mode flag for indirect or direct addressing).

Here is the RTL code for the CPU :

1. **Fetch**

1.1 FETCH 1 AR <- PC

1.2 FETCH 2 DR <- M[AR], PC <- PC+1

1.3 FETCH 3 IR <- DR[15...10], AR <- DR[9...0]

1. **Load (direct)**

2.1 LDAC1 DR <- M[AR]

2.2 LDAC2 AC <- DR

1. **Load (indirect)**

3.1 ILDAC1 DR <- M[AR]

3.2 ILDAC2 AC <- DR

3.3 ILDAC3 DR <- M[AR]

3.4 ILDAC4 AC <- DR

1. **Store AC content to memory**

4.1 STAC1 DR <- AC

4.2 STAC2 M[AR] <- DR

1. **Store AC content to memory (indirect)**

5.1 ISTAC1 DR <- M[AR]

5.2 ISTAC2 AR <- DR[9...0]

5.3 ISTAC3 DR <- AC

5.4 ISTAC4 M[AR] <- DR

1. **Increment AC**

6.1 INC AC <- AC+1, If (AC+1 == 0) Z <- 1 else Z <- 0

1. **Decrement AC**

7.1 DEC AC <- AC-1, If (AC+1 == 0) Z <- 1 else Z <- 0

1. **ADD**

8.1 ADD1 DR <- M[AR]

8.2 ADD2 AC <- AC+DR , If (AC+DR == 0) Z <- 1 else Z <- 0

1. **ADD (indirect)**

9.1 IADD1 DR <- M[AR]

9.2 IADD2 AR <- DR[9...0]

9.3 IADD3 DR <- M[AR]

9.4 IADD4 AC <- AC+DR, If (AC+DR == 0) Z <- 1 else Z <- 0

1. **SUB**

10.1 SUB1 DR <- M[AR]

10.2 SUB2 AC <- AC-DR, If (AC-DR == 0) Z <- 1 else Z <- 0

1. **SUB (indirect)**

11.1 ISUB1 DR <- M[AR]

11.2 ISUB2 AR <- DR[9...0]

11.3 ISUB3 DR <- M[AR]

11.4 ISUB4 AC <- AC-DR, If (AC-DR == 0) Z <- 1 else Z <- 0

1. **AND**

12.1 AND1 DR <- M[AR]

12.2 AND2 AC <- AC ^ DR , If (AC^DR == 0) Z <- 1 else Z <- 0

1. **AND (indirect)**

13.1 IAND1 DR <- M[AR]

13.2 IAND2 AR <- DR[9...0]

13.3 IAND3 DR <- M[AR]

13.4 IAND4 AC <- AC ^ DR, If (AC^DR == 0) Z <- 1 else Z <- 0

1. **OR**

14.1 OR1 DR <- M[AR]

14.2 OR2 AC <- AC v DR , If (AC v DR == 0) Z <- 1 else Z <- 0

1. **OR (indirect)**

15.1 IOR1 DR <- M[AR]

15.2 IOR2 AR <- DR[9...0]

15.3 IOR3 DR <- M[AR]

15.4 IOR4 AC <- AC v DR, If (AC v DR == 0) Z <- 1 else Z <- 0

1. **XOR**

16.1 XOR1 DR <- M[AR]

16.2 XOR2 AC <- AC **⊕** DR , If (AC **⊕** DR == 0) Z <- 1 else Z <- 0

1. **XOR (indirect)**

17.1 IXOR1 DR <- M[AR]

17.2 IXOR2 AR <- DR[9...0]

17.3 IXOR3 DR <- M[AR]

17.4 IXOR4 AC <- AC **⊕** DR, If (AC **⊕** DR == 0) Z <- 1 else Z <- 0

1. **NOT**

18.1 NOT AC <- AC **⊕** 1, If( AC **⊕** 1 == 0) Z <- 1 else Z <- 0

1. **Clear AC**

19.1 CLR AC <- 0, Z <- 1

1. **Shift Lef**t

20.1 SHL AC <- left shift AC

1. **Shift right**

21.1 SHR AC <- right shift AC

1. **Circular shift left**

22.1 CIL AC <- circular left shift AC

1. **Circular right shift**

23.1 CIR AC <- circular right shift AC

1. **Arithmetic shift left**

24.1 ASL AC <- arithmetic shift left AC

1. **Arithmetic shift right**

25.1 ASR AC <- arithmetic shift right AC

1. **Jump if carry (CF = 1)**

26.1 JC1 AR <- DR[9...0]

1. **Jump if not carry (CF = 0)**

27.1 JNC1 AR <- DR[9...0]

1. **Jump (unconditional)**

28.1 JMP1 AR <- DR[9...0]

1. **Jump if zero (ZF = 1)**

29.1 JZ1 AR <- DR[9...0]

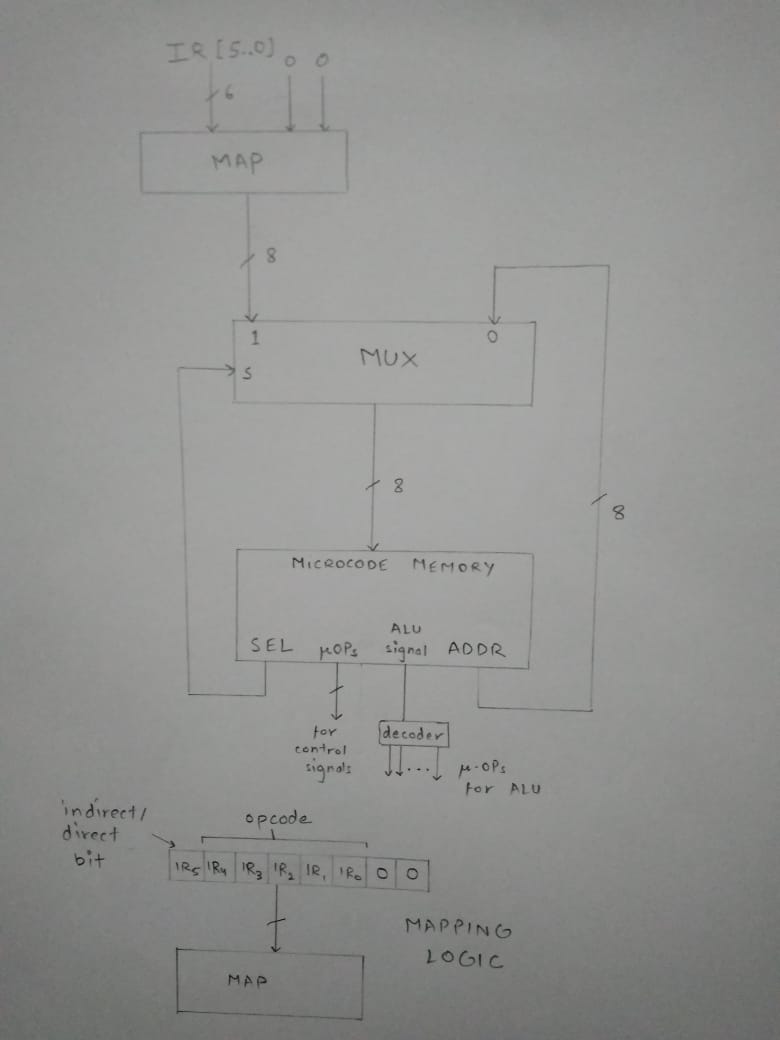
1. **Jump if not zero (ZF = 0)**

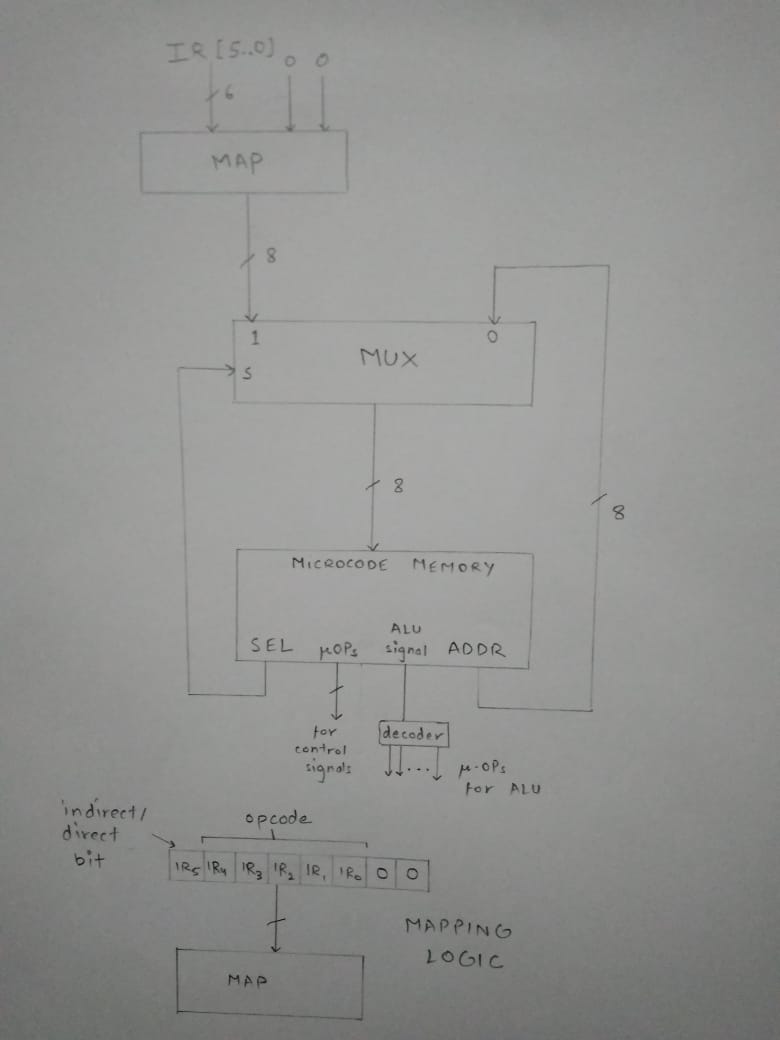
30.1 JNZ1 AR <- DR[9...0]

The control signals in the CPU are generated using a microprogrammed Control Unit implemented using the horizontal microcode. There are **13 control signals** which are generated using the horizontal microcode and some other ALU signals generated using vertical microcode simultaneously.

The CU consists of **61 states**.

The following is the **mapping logic** for the microsequencer:





The image above is the **microsequencer design** for the microprogrammed control unit.

**Microcode for the Control Unit**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STATE | address | sel | ARPC | DRM | PCINC | IRDR | ARDR | ACDR | MDR | DRAC | INC | DEC | CLR | PCDR | M3 | M2 | M1 | M0 | ADDR | |
| FETCH1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000001 | |
| FETCH2 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000010 | |
| FETCH3 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XXXXXXXX | |
| LDAC1 | 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000101 | |
| LDAC2 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| STAC1 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00001001 | |
| STAC2 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| ADD1 | 12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00001101 | |
| ADD2 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 00000000 | |
| SUB1 | 16 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00010001 | |
| SUB2 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 00000000 | |
| AND1 | 20 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00010101 | |
| AND2 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 00000000 | |
| OR1 | 24 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00011001 | |
| OR2 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 00000000 | |
| XOR1 | 28 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00011101 | |
| XOR2 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 00000000 | |
| JMPC1 | 32 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| JNC1 | 36 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| JMP1 | 40 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| JZ1 | 44 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| JNZ1 | 48 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| INC | 52 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| DEC | 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| NOT | 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 00000000 | |
| CLR | 64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| SHL | 68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 00000000 | |
| SHR | 72 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 00000000 | |
| CIL | 76 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 00000000 | |
| CIR | 80 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 00000000 | |
| ASL | 84 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 00000000 | |
| ASR | 88 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 00000000 | |
| HLT | 92 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01011100 | |
| ILDAC1 | 132 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10000101 | |
| ILDAC2 | 133 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10000110 | |
| ILDAC3 | 134 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10000111 | |
| ILDAC4 | 135 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| ISTAC1 | 136 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001001 | |
| ISTAC2 | 137 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001010 | |
| ISTAC3 | 138 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001011 | |
| ISTAC4 | 139 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00000000 | |
| IADD1 | 140 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001101 | |
| IADD2 | 141 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001110 | |
| IADD3 | 142 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10001111 | |
| IADD4 | 143 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 00000000 | |
| ISUB1 | 144 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010001 | |
| ISUB2 | 145 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010010 | |
| ISUB3 | 146 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010011 | |
| ISUB4 | 147 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 00000000 | |
| IAND1 | 148 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010101 | |
| IAND2 | 149 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010110 | |
| IAND3 | 150 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10010111 | |
| IAND4 | 151 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 00000000 | |
| IOR1 | 152 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011001 | |
| IOR2 | 153 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011010 | |
| IOR3 | 154 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011011 | |
| IOR4 | 155 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 00000000 | |
| IXOR1 | 156 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011101 | |
| IXOR2 | 157 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011110 | |
| IXOR3 | 158 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10011111 | |
| IXOR4 | 159 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 00000000 | |

The states M3,M2,M1,M0 correspond to the 4 bits of the vertically generated microcode for the ALU.

**Control Signal values** for the CPU

(the last signal ‘opType’ is an int which takes value from ‘ALU’ signal).

|  |  |
| --- | --- |
| **Signal** | **Value** |
| pcInc | PCINC |
| acInc | INC |
| acdec | DEC |
| acClr | CLR |
| pcOut | ARPC |
| arOut | 1 |
| drOut | DRM v IRDR v ARDR v ACDR v PCDR |
| acOut | DRAC |
| pcLoad | PCDR |
| drLoad | ARPC v ARDR |
| acLoad | DRAC v MDR |
| irLoad | ACDR v ALU(*m3m2m1m0*) |
| *(int)* opType | ALU(*m3m2m1m0*) |

**ALU Signals**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **M3** | **M2** | **M1** | **M0** | **Operation** |
| 0 | 0 | 0 | 1 | add |
| 0 | 0 | 1 | 0 | subtract |
| 0 | 0 | 1 | 1 | and |
| 0 | 1 | 0 | 0 | xor |
| 0 | 1 | 0 | 1 | not |
| 0 | 1 | 1 | 0 | or |
| 0 | 1 | 1 | 1 | Shift right |
| 1 | 0 | 0 | 0 | Shift left |
| 1 | 0 | 0 | 1 | Circular shift right |
| 1 | 0 | 1 | 0 | Circular shift left |
| 1 | 0 | 1 | 1 | Arithmetic shift right |
| 1 | 1 | 0 | 0 | Arithmetic shift left |