***Instruction Set Architecture:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code space(Binary) | Code Space (Hex) | Type | Number of Instructions | Note |
| **0000\_0000\_0000 – 0000\_1111\_1111** | **000 – 0FF** | **Special Instructions** | **256** | **Currently only NOP used , 255 free slots** |
| **0001\_0000\_0000 – 0001\_1111\_1111** | **100 – 1FF** | **Unconditional Jump** | **1** | **GOTO** |
| **0010\_0000\_0000 – 0011\_1111\_1111** | **200 – 3FF** | **ALU (Memory)** | **32** | **16 operations , 2 destinations** |
| **0100\_0000\_0000 – 0111\_1111\_1111** | **400 – 7FF** | **Conditional Jump** | **4** | **JC , JZ , JS , JO** |
| **1000\_0000\_0000 – 1111\_1111\_1111** | **800 - FFF** | **ALU(Immediate)** | **8** | **Currently 7 used , 1 free slot** |

1. Special Instruction :-

Syntax :-

|  |  |
| --- | --- |
| 0000 | Opcode(8 bits) |

Currently used NOP is used with the instruction code 0000\_0000\_0000

1. Unconditional Jump :-

Syntax :-

|  |  |
| --- | --- |
| 0001 | Immediate Address(8 bits) |

1. ALU instructions with memory content :-

Syntax :-

|  |  |  |  |
| --- | --- | --- | --- |
| 001 | d | Opcode(4 bits) | DMem index(4 bits) |

Here , “d” specifies the destination where the output of the ALU operation will be stored.

d = 1 , means results will be stored in accumulator , otherwise it will be stored in the memory location of the other operand.

Memory location of the other operand is given by the DMem index.

The operation is specified by Opcode.

|  |  |
| --- | --- |
| ***Opcode*** | ***Function*** |
| 0000 | Add memory element with accumulator |
| 0001 | Subtract accumulator by memory element |
| 0010 | Move value of accumulator to memory element |
| 0011 | Move value of memory element to accumulator |
| 0100 | Bitwise AND memory entry with accumulator |
| 0101 | Bitwise OR memory entry with accumulator |
| 0110 | Bitwise XOR memory entry with accumulator |
| 0111 | Subtract memory element by accumulator |
| 1000 | Increment memory entry |
| 1001 | Decrement memory entry |
| 1010 | Circular shift a memory entry left by the number of bits specified by accumulator |
| 1011 | Circular shift a memory entry right by the number of bits specified by the accumulator |
| 1100 | Shift a memory entry left by the number of bits specified by accumulator |
| 1101 | Shift a memory entry right by the number of bits specified by accumulator |
| 1110 | Arithmetic shifts a memory entry left by the number of bits specified by accumulator |
| 1111 | Complement the memory entry |

1. Conditional Jump :-

Syntax :-

|  |  |  |
| --- | --- | --- |
| 01 | Opcode(2 bits) | Immediate Address(8 bit) |

Jumps to the address depending on the status flag raised.

Opcode specifies the condition for jump.

|  |  |
| --- | --- |
| **Opcode** | **Function** |
| 00 | Jump to the address if Zero flag is raised |
| 01 | Jump to the address if Carry flag is raised |
| 10 | Jump to the address if Sign flag is raised |
| 11 | Jump to the address if Overflow flag is raised |

1. ALU instruction with immediate number :-

Syntax :-

|  |  |  |
| --- | --- | --- |
| 1 | Opcode(3 bits) | Immediate Number(8 bits) |

The operation with performed between the accumulator and the number given and stored in the accumulator.

The operation to be performed is given by the opcode.

|  |  |
| --- | --- |
| Opcode | Function |
| 000 | Add accumulator with immediate number |
| 001 | Subtract accumulator by the immediate number |
| 011 | Move immediate number to the accumulator |
| 100 | Bitwise AND immediate number with the accumulator |
| 101 | Bitwise OR immediate number with the accumulator |
| 110 | Bitwise XOR immediate number with the accumulator |
| 111 | Subtract accumulator from the immediate number |

