Weekly Report 3

Embedded Micro System Design (EEL 5930)

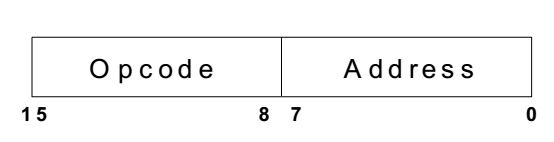
**Project : ASM2MIF Compiler Design**

**Author : Sourindu Chatterjee**

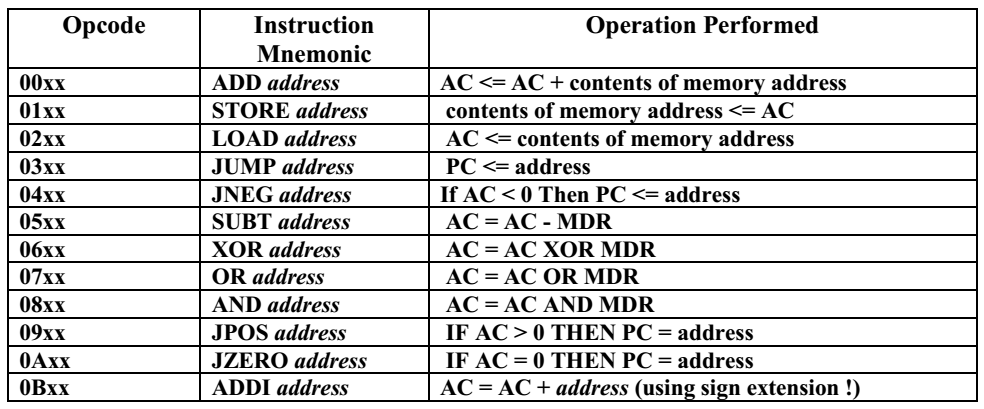
**Date : November 4, 2015**

|  |  |  |
| --- | --- | --- |
| Tasks | % of completion | % Of difficulty |
| Flex definations | **50%** | **25%** |
| Flex Rules | **50%** | **80%** |
| Flex User Code | **80%** | **10%** |
| Asm instructions | **80%** | **60%** |
| mIF instructions | **50%** | **50%** |

**MIF Instructions**

The Compiler design project that has to be studied is a special purpose compiler for a specific computer μp3 with Machine Instruction code of a specified order. The computer system executes a set of instructions, these instructions are stored in memory. The Machine Instruction Code or instruction set as we know it is divided into two major parts : Opcode & Address. The instruction set is 16 bits long, the low 16 bits contain a memory address field, where the high 16 bits are Opcode for the instruction which controls the flow of the program and tell the processor to perform specified operation on the data on the specified address filed. Depending on the opcode, this address may point to a data location or the location of another instruction. Here is figure to explain the same:

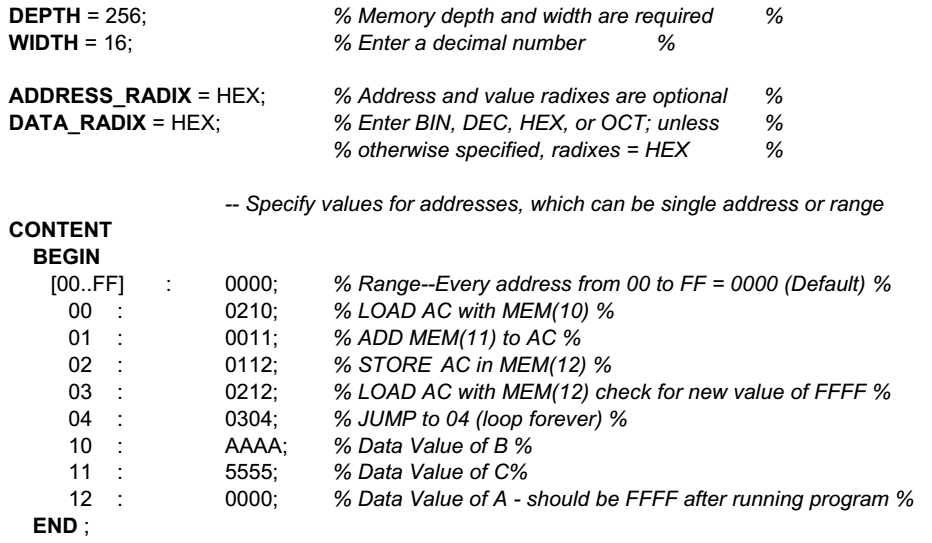
**Instruction Set :**



The instruction are not so complex but the ADDI is a bit complex because it deal with updating the program counter with the address in the Address code, which is only possible if the 16 bit value of the Address code is extended to 32 bits depending on the MSB (Most Significant Bit) and then adding the same to the program counter.

**Instruction Set :- B(AAAA) + C(5555)**

The executable steps for this set of program include three basic step for MIF instruction language, Firstly it sets all the data value for the Program Counter(Leftmost Column) to 0000, to Reset the data part. Secondly, it reads the first instruction and then read the data to the Accumulator (AC) depending on the Data Part (10) and map the instruction to the Instruction Part (02) for SCAN the data part to the Accumulator. Thirdly it executes a ADD instruction part (00) for memory location (11) that is data value of 5555. Fourthly it executes a PUSH instruction with opcode (01) and data code (12) which was previously having the value of 0000. Fifthly it executes a JMP instruction with opcode (03) and data code (04) because the PC (program counter) is also set at (04) at that current time it goes through a infinite loop cycle.



**Mapping the Opcode to Assembly for the typical stack machine would be :**

-- Label : L01

-- Label : L02

--Variable : B memory @ 10

--Variable : C memory @ 11

--Variable : O memory @ 12

|  |  |
| --- | --- |
| PUSH | B |
| PUSH | C |
| ADD |  |
| POP | O |
| JMP | L02 |

L01

|  |  |
| --- | --- |
| JMP | L02 |

L02