1. **Explain why, in MARIE, the MAR is 12 bits wide and the AC is 16 bit wide. (4 points)**

MARIE can handle 16-bit data, so the AC must be 16 bits wide. However, MARIE's memory is limited to 4096 address locations, so the MAR only needs to be 12 bits wide to hold the largest address.

1. **Write the MARIE assembly language equivalent for the machine instruction: 1011000000001111 and 1000100000000000 (6 points)**

At first, MARIE is a Machine Architecture that is Really Intuitive and Easy which is a system that consists of memory, CPU, Registers, and many more

This MARIE has nearly 13 instructions which are followed for its operations

Here in the above given 16 bits 4 bit is assigned for opcode that is operation instruction and remaining 12 bits is the address or condition bits

**1011000000001111**

Here there are 16 bits so let's divide it into 4+ 12

1011 is an opcode that represents MARIE instruction AddI X

Use the value present at address 000000001111 that is the address of X as the actual address and add the value to AC.

Here AddI is Add indirect.

00000000111

**1000100000000000**

Here there are 16 bits so let's divide it into 4+ 12

1000 is an opcode that represents MARIE instruction Skipcond

Skip the next instruction on condition by inspecting the value present in AC address 1000000000

1. **Combine Figure 4.12 (Fetch-Decode-Execute Cycle with interrupt Checking) and Figure 4.13 (Processing an Interrupt) to make one figure to show the whole process of Interrupt. Draw your own flowchart. (10 points)**



1. **Draw Figure 4.14 (a) (The Load 104 part of A Trace of the Program to Add Two Numbers) and explain each line about what is happening and why the registers have the value specified in the table. (10 points)**

The program is given below,

100 If, Load X /Load the first value

101 Subt Y /Subtract the value of Y, store result in AC

102 Skipcond 400 /If AC=0 (X=Y), skip the next instruction

103 Jump Else /Jump to Else part if AC is not equal to 0

104 Then, Load X /Reload X so it can be doubled

105 Add X /Double X

106 Store X /Store the new value

107 Jump Endif /Skip over the false, or else, part to the end of if

108 Else, Load Y /Start the else part by loading Y

109 Subt X /Subtract X from Y

10A Store Y /Store Y-X in Y

10B Endif, Halt /Terminate program (it doesn't do much!)

10C X, Dec 12 /Assume these values for X and Y 1 Comment Expert Answer

10D Y, Dec 20

1. **If we assemble the following MARIE assembly language program: (15 points)**

|  |  |
| --- | --- |
| **Memory Address (Hexadecimal)** | **Instruction** |
|  |  |
| **210** | **Load A** |
| **211** | **Sub B** |
| **212** | **Store C** |
| **213** | **Halt** |
| **214** | **0015** |
| **215** | **FFDE** |
| **216** | **0000** |
|  |  |

1. **Show the symbol table after the first pass.**

The instructions and their binary codes are listed in the Table 1 given below.



Table 1. MARIE's Instruction Set



The Table 2 given below shows the PC, IR, MAR, MBR and AC content after the execution of first three instructions.

Table 2 Register content after the execution of the first three instructions

The value at location Y is subtracted from the value at X and as the value at X is less than the value at Y, the result will be a negative number. The skip condition checks whether the result is zero or not. As the result is a nonzero value, the skip operation will not be done.

**2) Show the translated instructions after the first pass.**

In this part, a new instruction needed to be added (Jumpl Offset X). Here the offset address value is stored at location X. So first, the content of X need to be read from memory to MBR and add it with AC and then place it on to the PC. By assuming the opcode fetch is done,the RTN for Jumpl Offset X is given below.

 MAR ← X

MBR ← M

[MAR] AC ← AC+ MBR

PC ← AC

**3) Show the translated instructions after the second pass.**

In this part, the program for reading 8 integers as input and finding out the largest and smallest value is done.

|  |  |  |  |
| --- | --- | --- | --- |
| Label | Instruction | Operand | Comment |
|  | ORG 100 |  | /PROGRAM STARTS AT LOCATION 100 |
| LOOP, | LOAD | X |  |
|  | SUBT | EIGHT |  |
|  | SKIPCOND | 000 | / CHECKS WHETHER OUR ALL DATA ARE INPUTED OR NOT |
|  | JUMP | PRINT |  |
|  | CLEAR |  |  |
|  | INPUT |  |  |
|  | STORE | TEMP |  |
|  | CLEAR |  |  |
|  | LOAD | X | /CHECK WHETHER THE INPUTTED DATE IS THE FIRST ONE OR NOT |
|  | SUBT | ZERO |  |
|  | SKIPCOND | 800 |  |
|  | JUMp | STOREFIRST |  |
|  | CLEAR |  | /FIND WHETHER THE CURRENT DATA IS LESS THAN MIN OR NOT |
|  | LOAD | TEMp |  |
|  | SUBT | MIN |  |
|  | SKIPCOND | 000 |  |
|  | JUMP | FINDMAX |  |
|  | JUMP | MINSTORE |  |
| CONTINUE, | CLEAR |  | /INCREMENT THE DATA COUNTER |
|  | LOAD | X |  |
|  | ADD | ONE |  |
|  | STORE | X |  |
|  | CLEAR |  |  |
|  | JUMP | LOOP |  |
| FINDMAX, | CLEAR |  | /FIND WHETHER THE CURRENT DATA IS GREATER THAN MAX OR NOT |
|  | LOAD | TEMP |  |
|  | SUBT | MAX |  |
|  | SKIPCOND | 800 |  |
|  | JUMP | CONTINUE |  |
|  | JUMP | MAXSTORE |  |
| MINSTORE, | CLEAR |  | /STORE THE MINIMUM VALUE TO MIN |
|  | LOAD | TEMP |  |
|  | STORE | MIN |  |
|  | JUMP | CONTINUE |  |
| MAXSTORE, | CLEAR |  | /STORE THE MAX VALUE TO MAX |
|  | LOAD | TEMP |  |
|  | STORE | MAX |  |
|  | JUMP | CONTINUE |  |
| STOREFIRST, | LOAD | TEMP | / STORE THE FIRST DATA ON TO MAX AND MIN LOCATIONS |
|  | STORE | MIN |  |
|  | STORE | MAX |  |
|  | JUMp | CONTINUE |  |
| PRlNT, | LOAD | MAX | /PRINT THE MAXIMUM AND MINIMUM VALUE |
|  | OUTPUT |  |  |
|  | CLEAR |  |  |
|  | LOAD | MIN |  |
|  | OUTPUT |  |  |
|  | CLEAR |  |  |
| EIGHT, | HALT |  |  |
| ONE, | DEC 8 |  | /LOCATION FOR CONSTANT VALUE 8 TO CHECK THENUMBER OF INPUT |
| ZERO, | DEC 1 |  | /LOCATION FOR CONSTANT VALUE 1 TO INCREMENT THE COUNTER |
| x, | DEC O |  | /LOCATION FOR CONSTANT VALUE O |
| MIN, | DEC O |  | /LOCATION FOR COUNT OF INPUT DATA |
| MAX, | DEC O |  | /LOCATION FOR MINIMUM VALUE |
| T EMR | DEC O |  | /LOCATION FOR MAXIMUM VALUE |
| EIGHT, | DEC O |  | /TEMPORARY LOCATION FOR STORING CURRENT INPUT |

In the above program, eight different inputs are taken and stored to a temporary location called TEMP. Every time the content of TEMP is compared with MAX and MIN. If TEMP content is found to be greater than MAX then MAX content is replaced with TEMP content. If TEMP content is less than MIN content, the MIN content is replaced with TEMP Content. The data count is done by X and every time X is incremented when a new data is inputted and once it reaches 8, the MIN and MAX will have smallest and largest value respectively. The program then outputs the MAX and MIN value. The screenshots of the simulation is shown below.



The above screenshot shows the input values given while running the code.



The above screenshot shows the output obtained after the simulation. The maximum value and minimum values are listed at the output screen.

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..