**TEST PROGRAMS**

Below, you will find 4 test program written using the assemble language code introduced earlier. In the first test program, we will test the arithmetic operations and other related operations such as input, output. In the second program I will test the branching/jumping instruction. In the third program I will be testing the subroutine. The fourth test programs will test the Makeable Hard Vectored Priority Interrupt System. More details will be given later.

**First Test Program**

In this first test program we will be testing we will employ a straight line flow through the program. We will be testing arithmetic instruction such as add, subtract, multiply, etc. We will also be testing the load/store instruction. And the input/output instruction will also be tested. All of the three addressing mode will be used in this test program. More explanations will be given throughout the programs. After executing the program we can see that the program has passed the test successfully because in the last 4 cycle of the test program we can see that the last 4 value in the Accumulator will be as follow: 3, 13, 6, 3 and the value in the bus line when the program ended should be 3. This will verify that the processor is working correctly.

|  |  |  |
| --- | --- | --- |
| Assembly Codes | Explanations | Machine Codes |
| LOAD 10  MUL 7  STR 100  LOAD 90  DIV 30  OUTPUT  STR 70  LOAD 29  SUB @70  SHIFTR  STR 71  INPUT  ADD #100  OR 6  STR 72  LOAD @70  LOAD @71  LOAD @72  LOAD #100 | Load the accumulator with 10.  Multiply the accumulator with 7, therefore the value in the accumulator is now 70.  Store the value in the accumulator in the DRAM address 100, hence the address 100 in the DRAM has the value 70.  ACC=90  ACC=30/100=3  Output the value in the ACC to Bus, therefore BUS=3  Store value from ACC into DRAM address 70, therefore MEM[70]=3  ACC=29  ACC=29-MEM[70]=29-3=26  Right shift the value in the accumulator,  since ACC was 26, then ACC>>1 is 13  Store ACC is address 71, hence MEM[71]=13  Input the value from the Bus into the ACC.  Since BUS=3, then ACC=3.  This is add indirect so, the value at address 100 is 70, and the value at address 70 is 3.  Therefore, ACC=3+MEM[MEM[100]]  ACC=3+MEM[70]=3+3=6  So the ACC value is now 6.  Since ACC= 6 then ACC>>6|6=6  Store the result in MEM[72]  ACC=MEM[70]=3  ACC=MEM[71]=13  ACC=MEM[72]=6  ACC= MEM[MEM[100]]=MEM[70]=3 | 01000 000 00001010  11000 000 00000111  01001 000 01100100  01000 000 01011010  11001 000 00011110  01101 000 00000000  01001 000 01000110  01000 000 00011101  00001 001 01000110  01011 000 00000000  01001 000 01000111  01100 000 00000000  00000 010 01100100  00010 000 00000110  01001 000 01001000  01000 001 01000110  01000 001 01000111  01000 001 01001000  01000 010 01100100 |

**Second Test Program**

In this second we will be testing the branching/jumping instruction. We will start with straight line control flow. After that we will go through a looping structure. Since we will be going through a loop 14 times, we should be able to see that the output on the Bus should be decreasing by 1 each time we go through the loop. Therefore, to confirm that this program ran correctly, on the bus we should see these values: 14,13,12,11,10,9,…,3,2,1,0.

|  |  |  |
| --- | --- | --- |
| Assembly Codes | Explanations | Machine Codes |
| LOAD 7  BRANCH 5  ADD 1  ADD 1  STR 20  Address 5:  ADD 3  STR 20  LOAD 14  Address 8:  SUB 1  OUTPUT  BRZERO 12  BRANCH 8  Address 12:  ADD 222  OUTPUT  LOAD @20 | Load the ACC with 7  Branch to address 5.  These instruction should not be executed because we branched to address 5, and skipped these three instruction.  After branching here, we will add the accumulator with 3.  Therefore, ACC = ACC+3 = 7+3=10  Store the results in memory location 20.  ACC= 14  ACC=ACC-1  Bus=ACC  If the zero flag is set, we will exit the loop by branching to address 12. Else, we will continue.  We will branch back to address 8, and perform the subtraction loop again.  We will branch here after the zero flag is set.  ACC=ACC+222=0+222=222  Bus=ACC  To verify that the normal branch we perform earlier was correct, we should be getting the value 10 in the ACC after we loaded it. | 01000 000 00000111  11010 000 00000101  00000 000 00000001  00000 000 00000001  01001 000 00010100  00000 000 00000011  01001 000 00010100  01000 000 00001110  00001 000 00000001  01101 000 00000000  11011 000 00001100  11010 000 00001000  01000 000 11011110  01101 000 00000000  01000 001 00010100 |

**Third Test Program**

In this third test program we will be testing the Jump Subroutine instruction, and the return from subroutine ability. After reading the explanations, we can see that the program ran successfully if the memory location 55 contains the value 33, and memory location 56 contains value 3.

|  |  |  |
| --- | --- | --- |
| Assembly Codes | Explanations | Machine Codes |
| LOAD 15  ADD 2  JUMPSUB 43  LOAD 11  MUL 3  STR 56  LOAD 0  LOAD @55  OUTPUT  LOAD @56  OUTPUT  Address 43:  ADD 2  ADD 2  DIV 7  STR 55  RETURNSUB | ACC=15  ACC=15+2=17  Jump to instruction address 43  These codes will be run after we return from the subroutine.  ACC =11  ACC = 11\*3=33  MEM[56]=ACC  ACC=0  ACC=MEM[55]  Bus=ACC  ACC=MEM[56]  Bus=ACC  After jumping here we will run these subroutine codes.  ACC=ACC+2=15+2=19  ACC=19+2=21  ACC=21/7=3  MEM[55]=ACC  Return to the instruction address 4.  And continue executing the program. | 01000 000 00001111  00000 000 00000010  11100 000 00101011  01000 000 00001011  11000 000 00000011  01001 000 00111000  01000 000 00000000  01000 001 00110111  01101 000 00000000  01000 001 00111000  01101 000 00000000  00000 000 00000010  00000 000 00000010  11001 000 00000111  01001 000 00110111  11101 000 00000000 |

**Fourth Test Program**

In this test program we will be testing the MHVPI System. In our processor there will be 4 interrupt program. Those interrupt are: overflow interrupt, illegal instruction interrupt, illegal addressing mode interrupt, and external interrupt. The interrupt service for these programs will start from the following address: 100, 110, 120, and 130. After executing the interrupt we will call RETURNITR to go back and continue executing the program. We can verify that the program ran correctly because the last three value on the Bus will be 1,2,3.

|  |  |  |
| --- | --- | --- |
| Assembly Codes | Explanations | Machine Codes |
| LOAD 115  SUB 209  ADD 0  MASKHVPI 0010  #$%@#&@@#\*&@!$  LOAD 0  MASKHVPI 0100  #$%@#&@@#\*&@!$  LOAD 0  LOAD @90  OUTPUT  LOAD @91  OUTPUT  LOAD @92  OUTPUT  Address 100:  LOAD 1  STR 90  OUTPUT  RETURNITR  Address 110:  LOAD 2  STR 91  OUTPUT  RETURNITR  Address 120:  LOAD 3  STR 92  OUTPUT  RETURNITR | ACC=115  ACC=115-209  This will cause an over low. Therefore, the interrupt will be enable. And we will jump to the overflow interrupt which is at address 100.  ACC=ACC+0  Mask the Interrupt Priority Register with the value 0010, therefore, this will give the illegal instruction interrupt the highest priority.  This is an illegal instruction, and illegal addressing mode. However, we give the priority to the illegal instruction, therefore we will jump to address 110, to execute the interrupt service for illegal instruction.  These code will be run after we return from the illegal instruction interrupt service.  ACC=0  Mask the Interrupt Priority Register with the value 0100, therefore, this will give the illegal addressing mode interrupt the highest priority.  This is an illegal instruction, and illegal addressing mode. However, we give the priority to the illegal instruction, therefore we will jump to address 120, to execute the interrupt service for illegal instruction.  These code will be run after we return from the illegal instruction interrupt service.  ACC=0  ACC= MEM[90]  Bus=ACC  ACC= MEM[91]  Bus=ACC  ACC= MEM[92]  Bus=ACC  These are the codes for the overflow interrupt service:  ACC=1;  MEM[90]=ACC  Bus=ACC  Return to the main program.  These are the codes for the illegal instruction service:  ACC=2;  MEM[90]=ACC  Bus=ACC  Return to the main program.  These are the codes for the illegal addressing mode service:  ACC=3;  MEM[90]=ACC  Bus=ACC  Return to the main program. | 01000 000 01110011  00001 000 11010001  00000 000 00000000  01111 000 00000010  11111 111 01100000  01000 000 00000000  01111 000 00000100  11111 111 11100000  01000 000 00000000  01000 001 01011010  01101 000 00000000  01000 001 01011011  01101 000 00000000  01000 001 01011100  01101 000 00000000  Address 100:  01000 000 00000001  01001 000 01011010  01101 000 00000000  11110 000 00000000  Address 110:  01000 000 00000010  01001 000 01011011  01101 000 00000000  11110 000 00000000  Address 120:  01000 000 00000011  01001 000 01011100  01101 000 00000000  11110 000 00000000 |