PROJECT REPORT

Single Cycle and Pipelined Processor

Contents

1. Design and Implementation ............................................................................................................................................ 2 a. Design detailed description ........................................................................................................................................ 2
   1. Component circuits drawings ..................................................................................................................................... 2

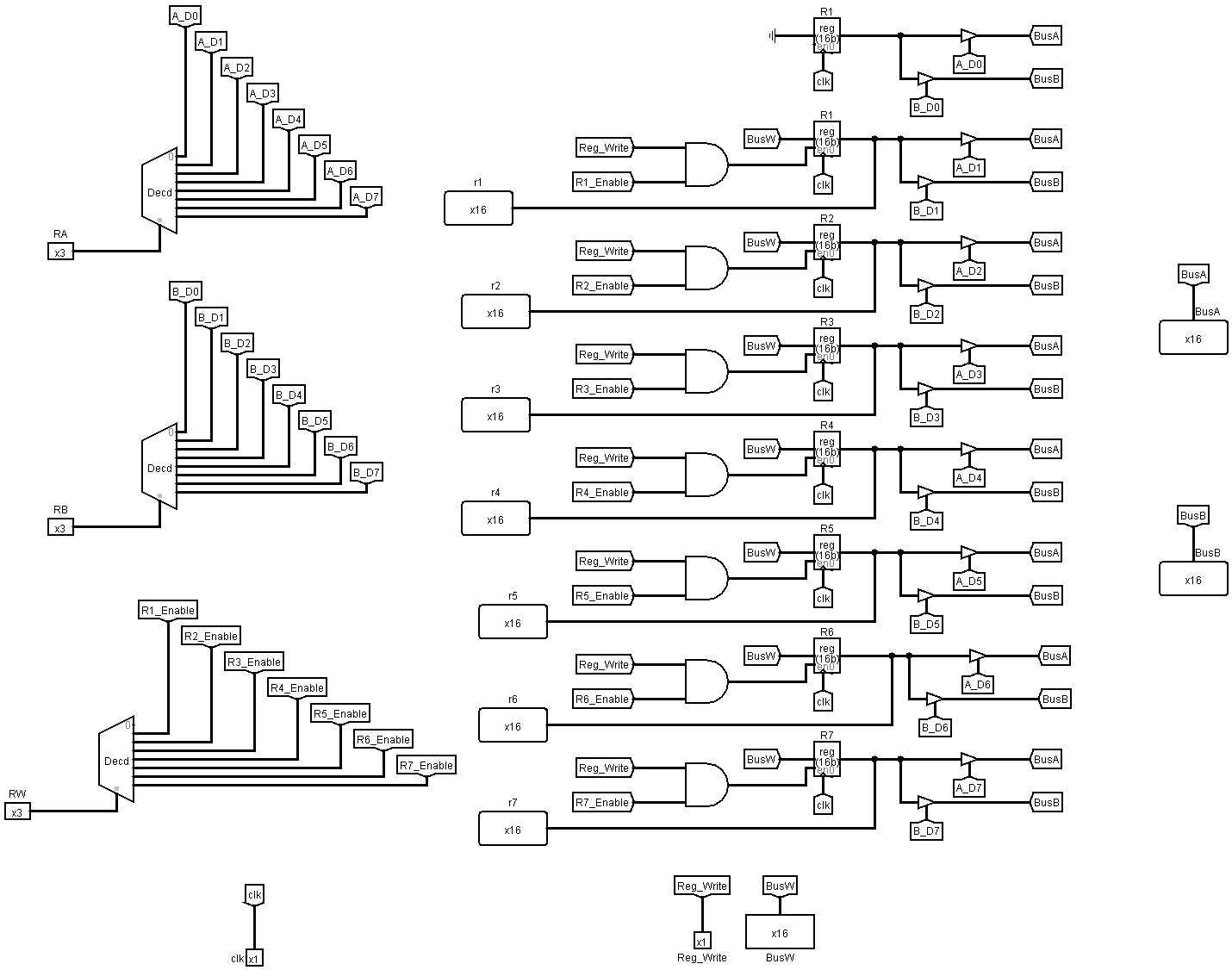
i. Register File ............................................................................................................................................................. 2 ii. ALU .......................................................................................................................................................................... 3 iii. Logical Unit .............................................................................................................................................................. 3 iv. Shifter ...................................................................................................................................................................... 4 v. AU ............................................................................................................................................................................ 4 vi. LU and Shifter .......................................................................................................................................................... 5 vii. Main controller ................................................................................................................................................... 5 viii. PC ........................................................................................................................................................................ 6 ix. Next PC .................................................................................................................................................................... 6 x. CPU(data path) ........................................................................................................................................................ 7 xi. NPC .......................................................................................................................................................................... 7 xii. P2 ........................................................................................................................................................................ 8 xiii. P3 ........................................................................................................................................................................ 8 xiv. Ex MEM WB ......................................................................................................................................................... 9 xv. HAZARD DETECT AND FORWARD ..................................................................................................................... 10 c. Complete description of the control logic and control signals with a table and logic equations ............................. 11

i. Signals control table .............................................................................................................................................. 11 ii. Logic equations ..................................................................................................................................................... 12 d. Description of the forwarding logic, and the logic implemented in stalling ............................................................. 12

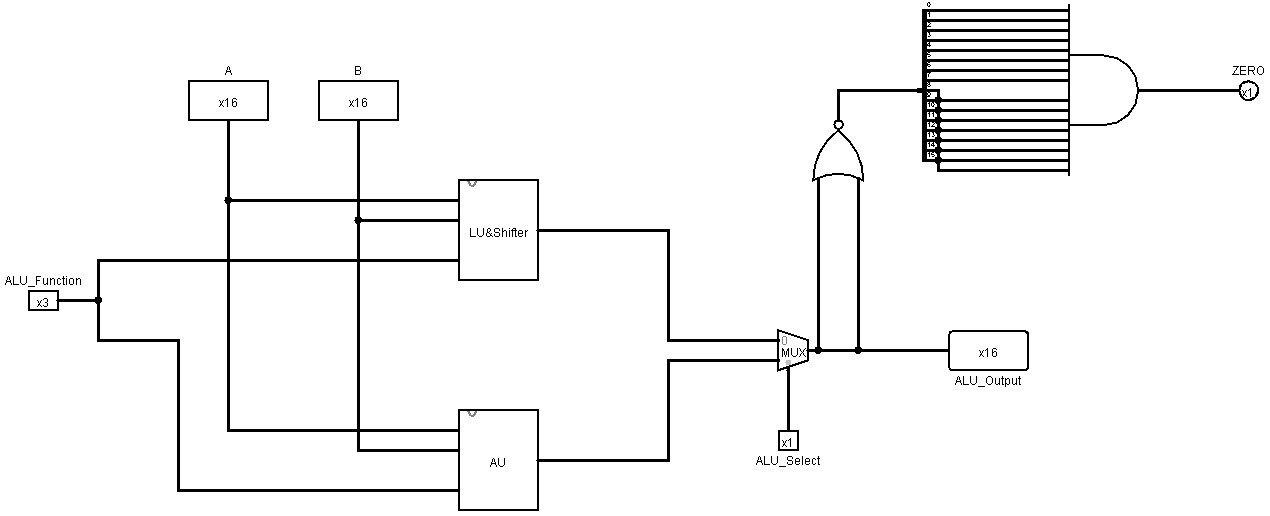
1. Simulation and Testing Programs ................................................................................................................................. 13 a. Number of ones in a register .................................................................................................................................... 13
   1. Selection sort procedure: .......................................................................................................................................... 14
   2. Remaining untested instructions: ............................................................................................................................. 15

## b. Component circuits drawings

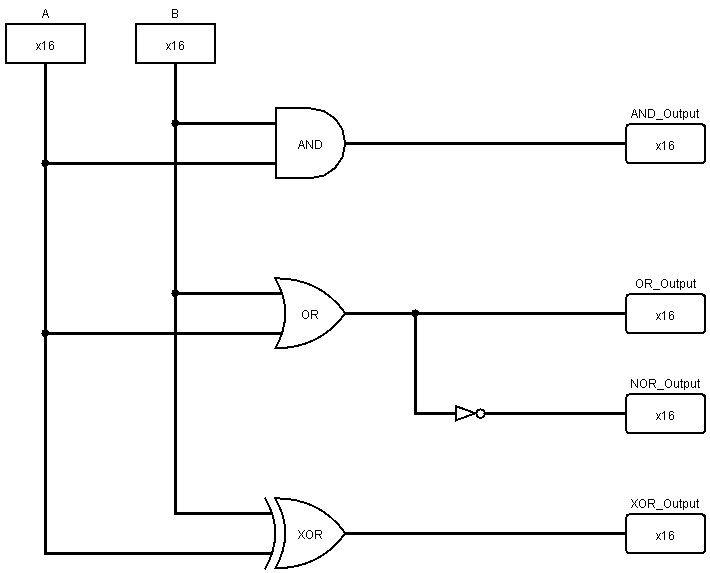
### i. Register File



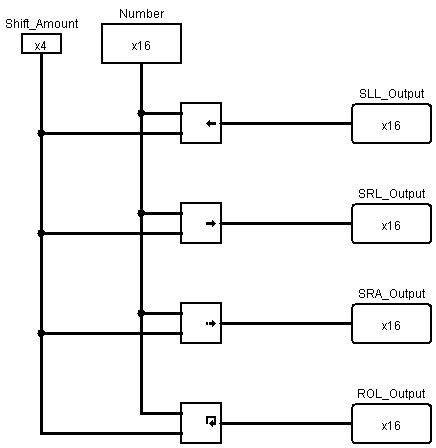
### ii. ALU



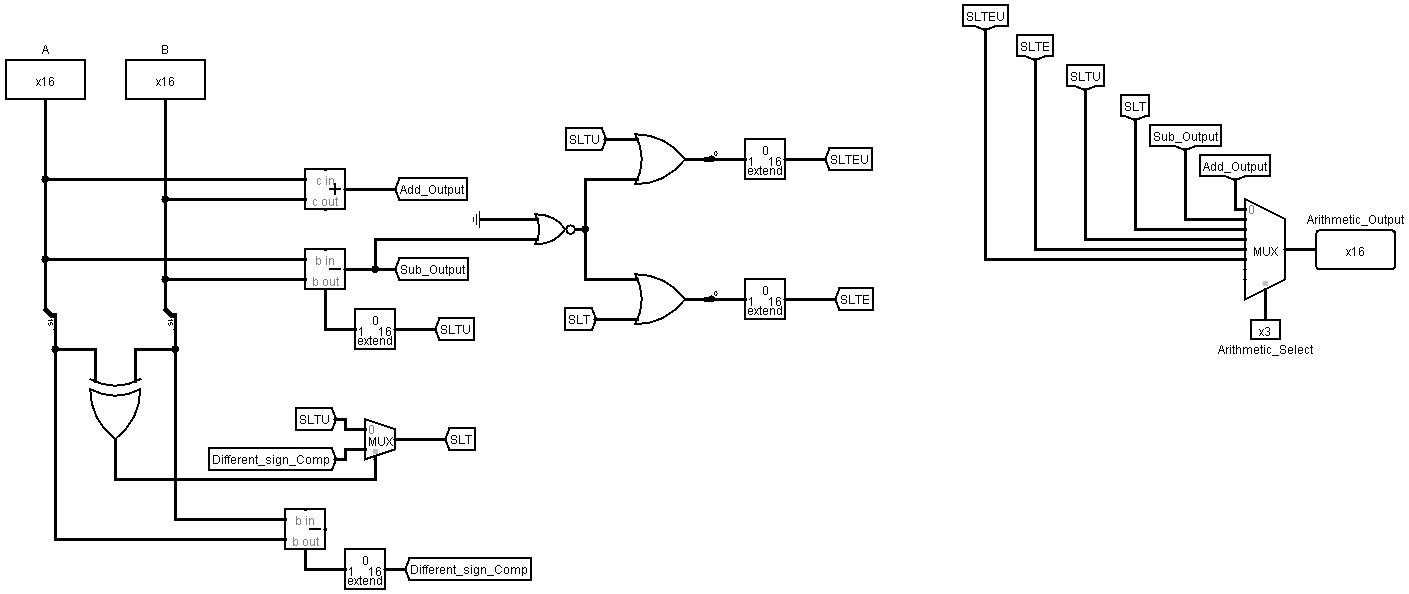
### iii. Logical Unit



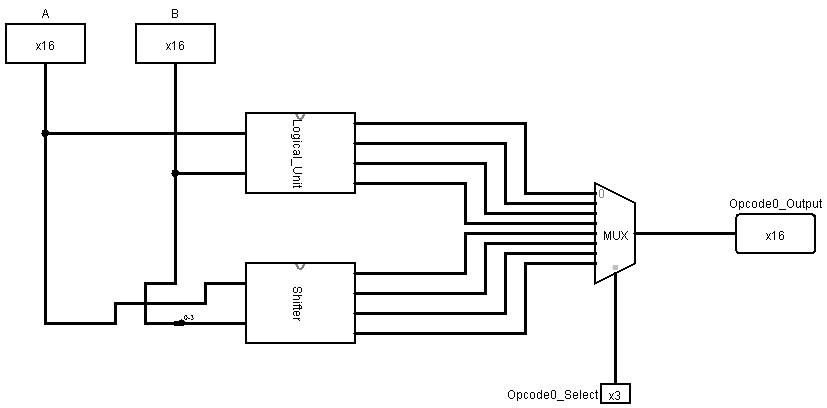
### iv. Shifter



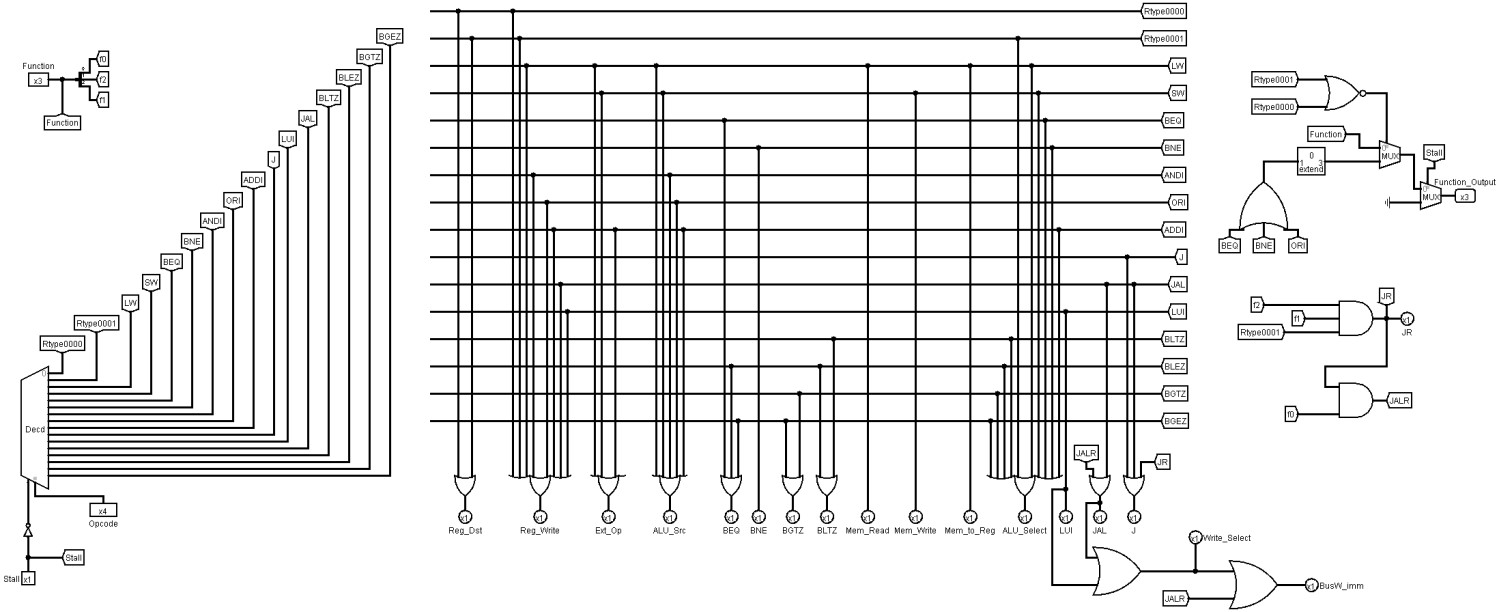
### v. AU



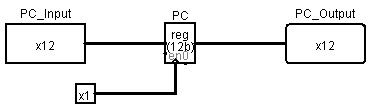
### vi. LU and Shifter



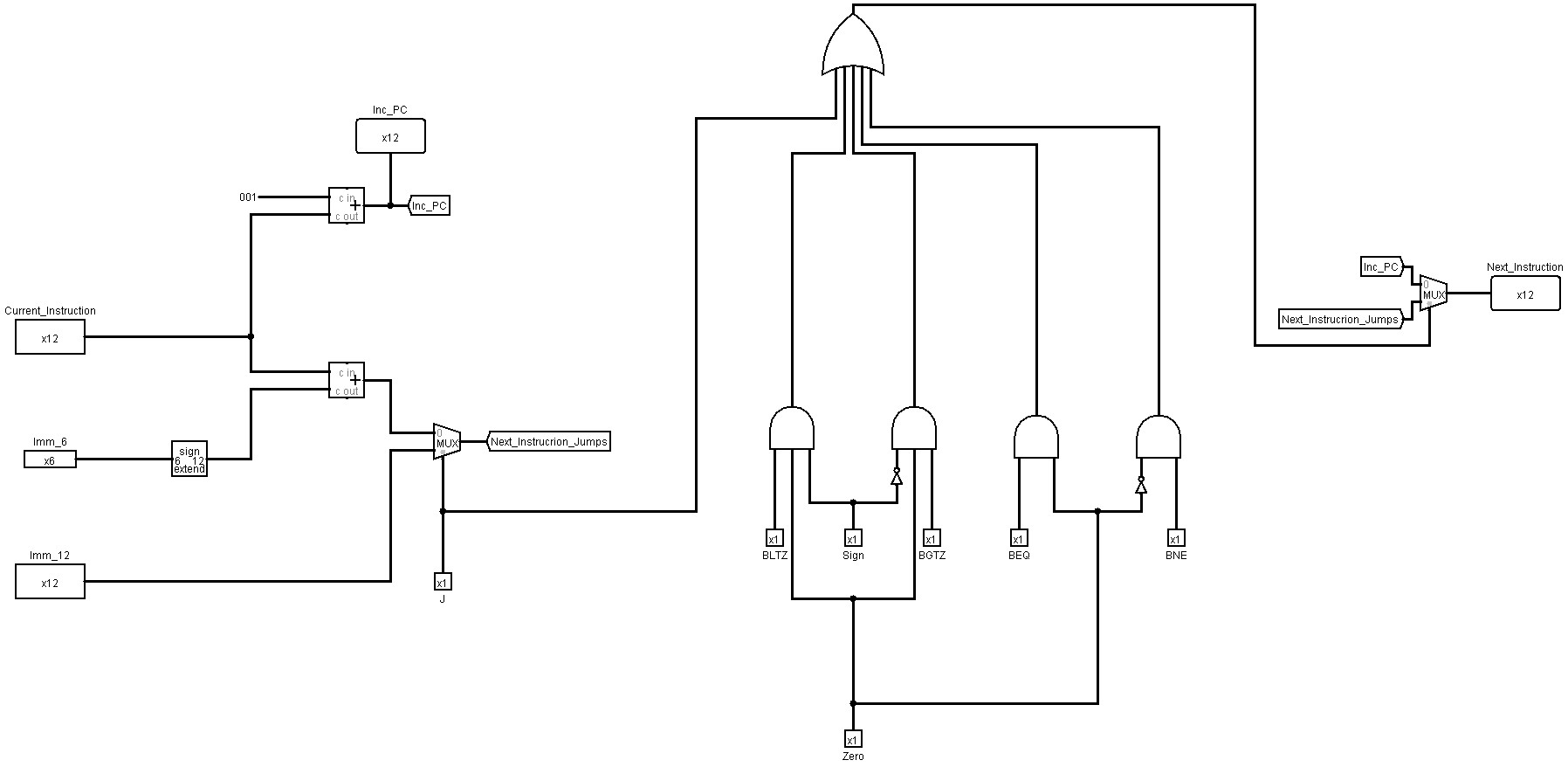
### vii. Main controller



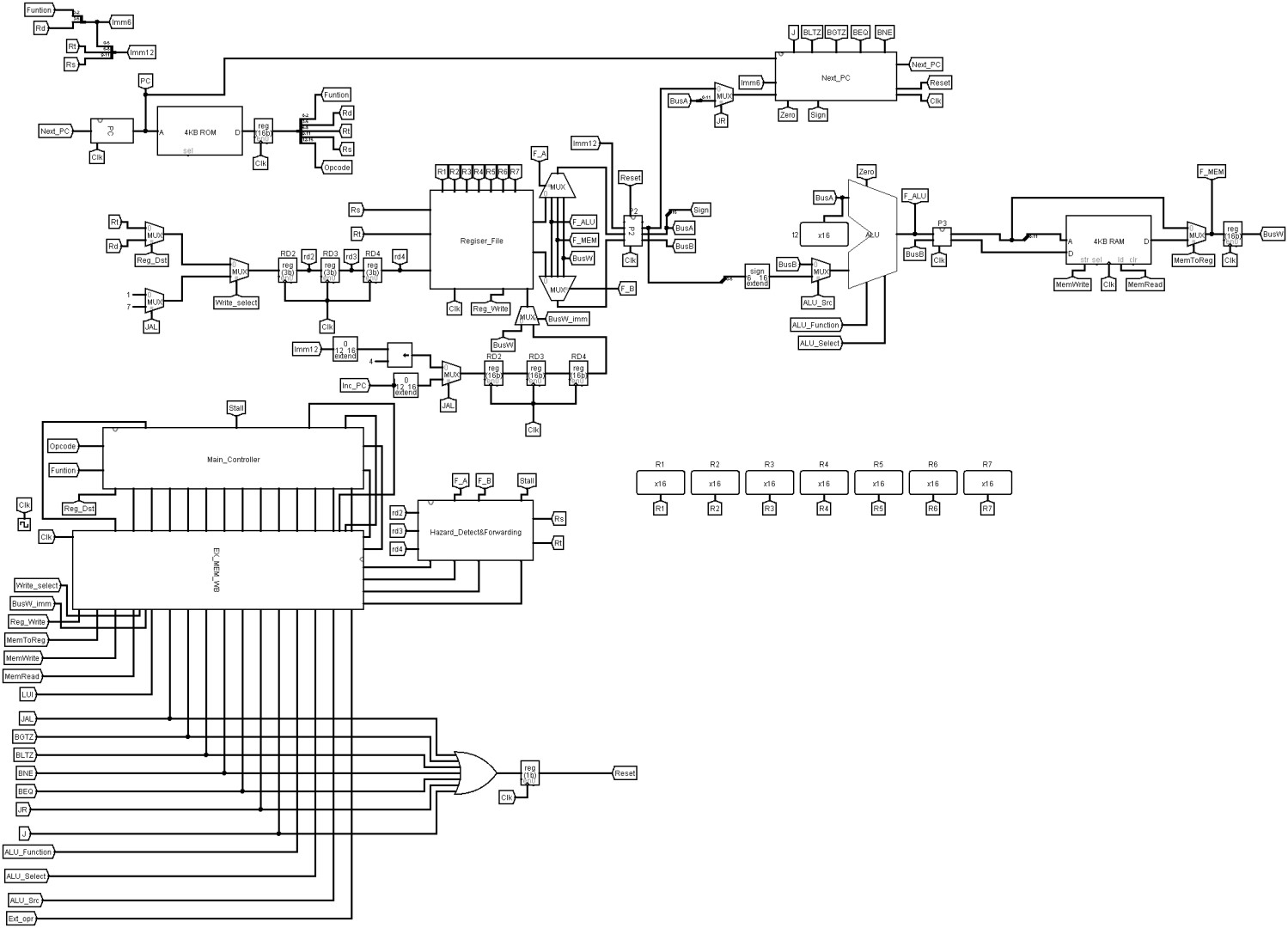
### viii. PC



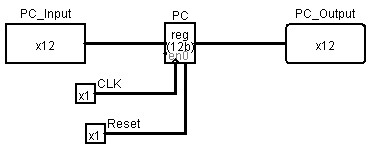
### ix. Next PC



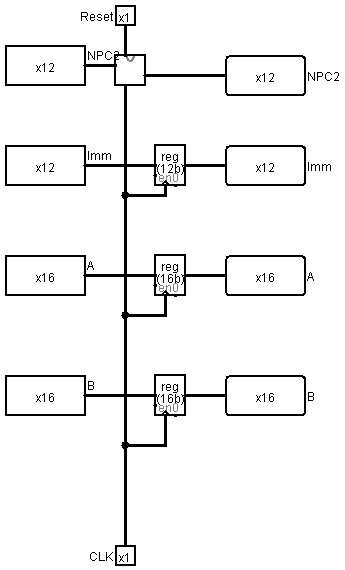
### x. CPU(data path)



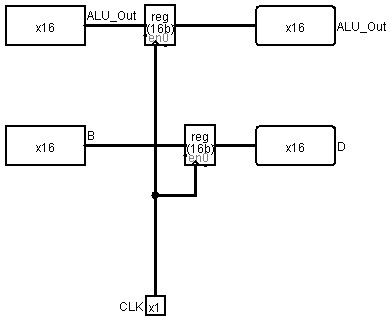
### xi. NPC



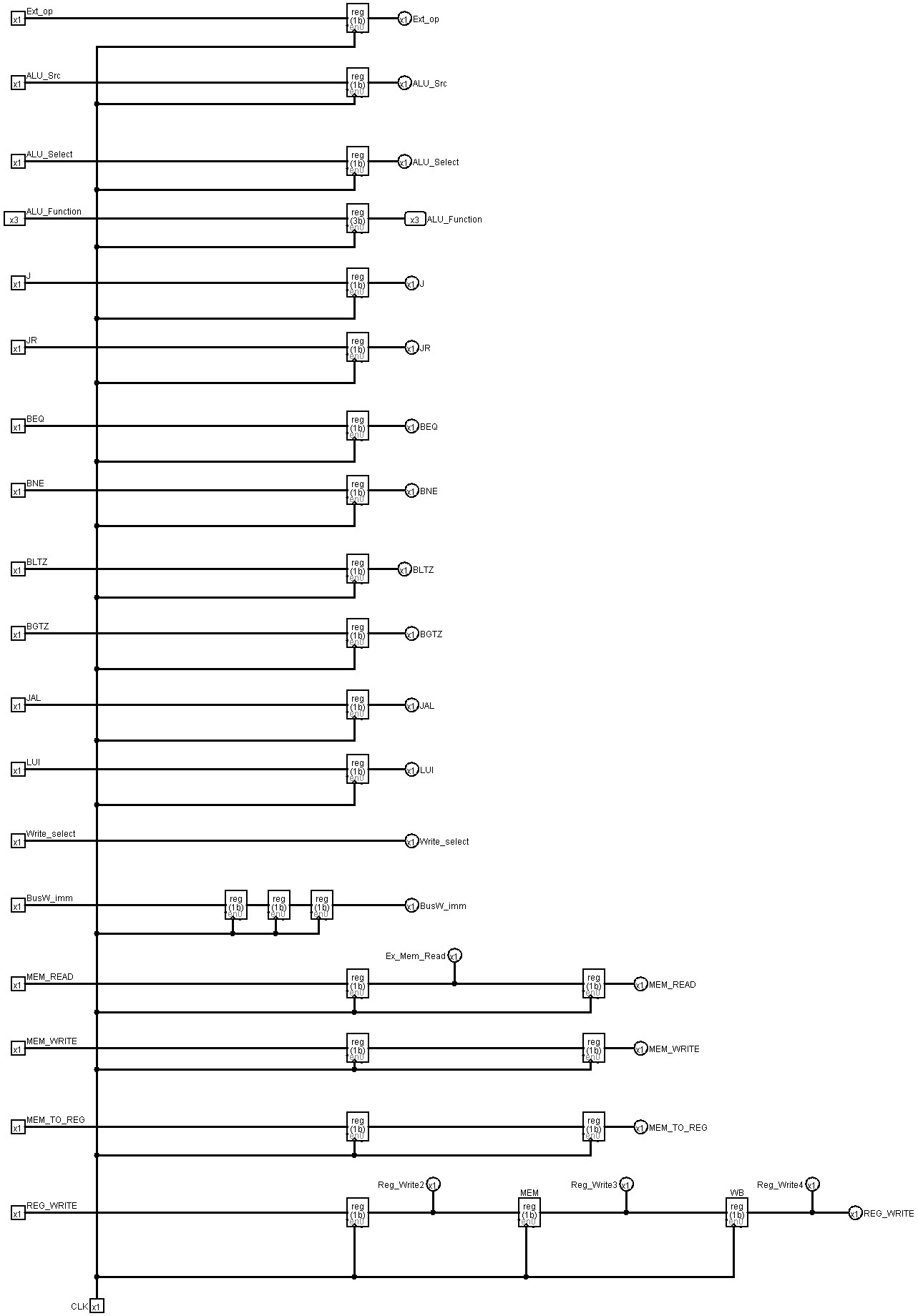
### xii. P2



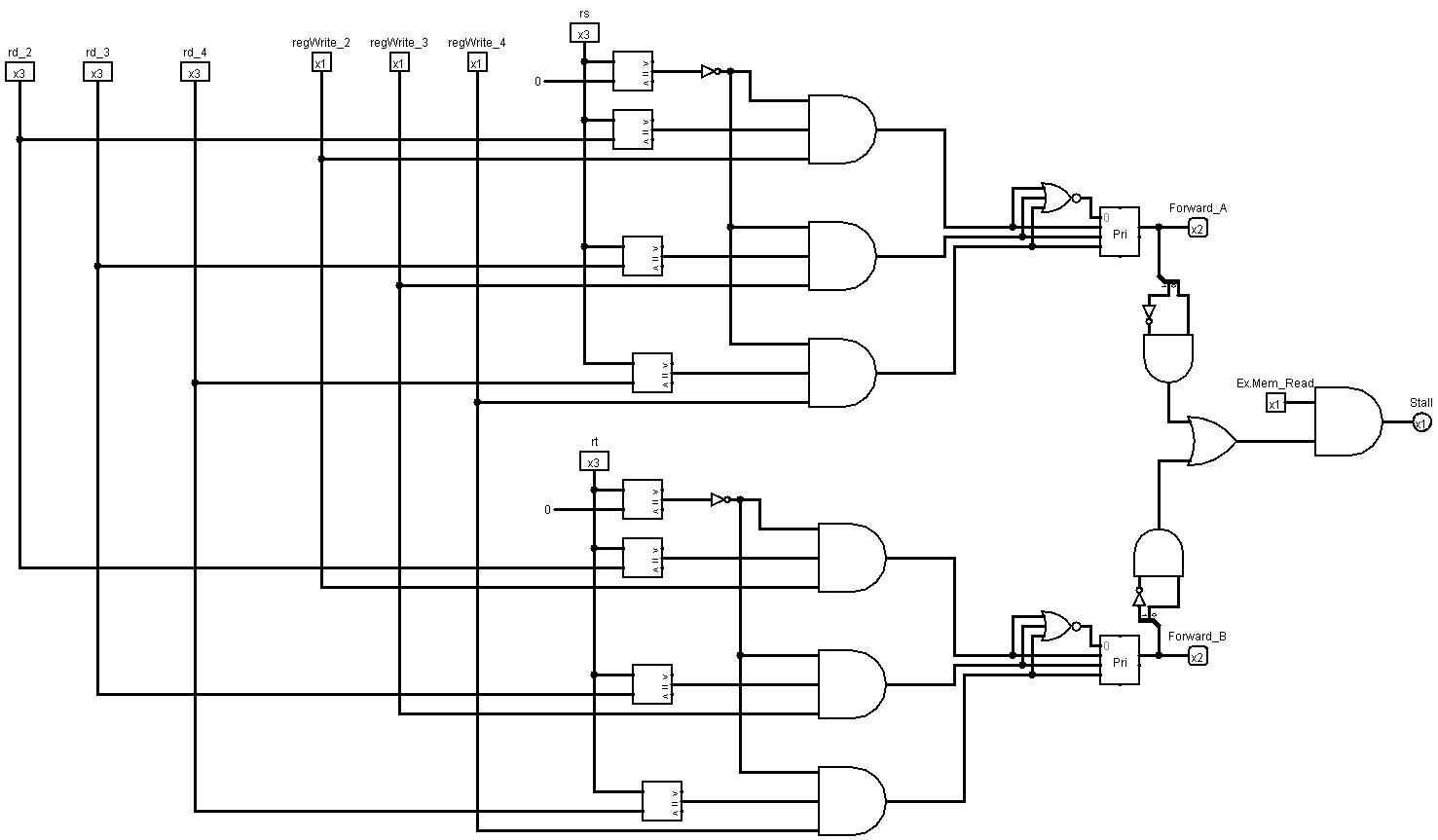
### xiii. P3



### xiv. Ex MEM WB



### xv. HAZARD DETECT AND FORWARD



## c. Complete description of the control logic and control signals with a table and logic equations

### i. Signals control table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Reg** | **Reg** | **Ext** | **ALU** | **ALU** | **Beq** | **Bne** | **BGZ** | **BLZ** | **J** | **JAL** | **LUI** | **Mem** | **Mem** | **Mem** | **ALU** |
|  |  |  |  |  |  |  |  |
|  | **Dst** | **Write** | **Op** | **Src** | **Op** |  |  |  |  |  |  |  | **Read** | **Write** | **toReg** | **Select** |
| **R-**  **type** | **1 = Rd** | **1** | **x** | **0=BusB** | **R-**  **type** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **x** |
| **lw** | **0 = Rt** | **1** | **1=sign** | **1=Imm** | **ADD** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **1** |
| **sw** | **x** | **0** | **1=sign** | **1=Imm** | **ADD** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **x** | **1** |
| **beq** | **x** | **0** | **x** | **0=BusB** | **SUB** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |
| **bne** | **x** | **0** | **x** | **0=BusB** | **SUB** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |
| **andi** | **0 = Rt** | **1** | **0=zero** | **1=Imm** | **AND** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **ori** | **0 = Rt** | **1** | **0=zero** | **1=Imm** | **OR** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **addi** | **0 = Rt** | **1** | **1=sign** | **1=Imm** | **ADD** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **j** | **x** | **0** | **x** | **x** | **x** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **x** | **0** |
| **jr** | **x** | **0** | **x** | **x** | **x** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| **lui** | **x** | **1** | **x** | **1=Imm** | **SLL** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| **jal** | **x** | **1** | **x** | **x** | **x** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **x** | **0** |
| **jalr** | **x** | **1** | **0** | **0** | **x** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **bltz** | **x** | **0** | **x** | **0=BusB** | **SUB** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |
| **blez** | **x** | **0** | **x** | **0=BusB** | **SUB** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |
| **Bgtz** | **x** | **0** | **x** | **0=BusB** | **SUB** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |
| **Bgez** | **x** | **0** | **x** | **0=BusB** | **SUB** | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **x** | **1** |

### ii. Logic equations

**Reg Dst <= R-type**

**Reg Write <= R-type + lw + sw + andi + ori + addi + jal + lui**

**Ext Op <= lw + sw + addi**

**ALU\_Src <= lw + sw + andi + ori + addi**

**BEQ <= beq + blez + bgez**

**BNE <= bne**

**BGTZ <= bgtz + bgez**

**BLTZ <= bltz + blez**

**Mem Read <= lw**

**Mem Write <= sw**

**Mem to Reg <= lw**

**ALU Select <= R-type + lw + sw + beq + bne + addi + lui + bgtz + bgez + blez + bltz**

**LUI <= lui**

**JAL <= jalr + jal**

**J <= jr + jal + j**

## d. Description of the forwarding logic, and the logic implemented in stalling

forwarding logic:

first we check for dependencies in rd, or rt, then if there is a dependency we take the rd or rt value from that stage, then the value is forwarded.

stalling logic:

if there is any jump related instruction (j, branch), we save its address until we reach the ALU stage, then we jump

and hold the other instructions for a certain cycles delay, after that we continue executing the instructions.

# 2.Simulation and Testing Programs

## a. Number of ones in a register

This test's purpose is to count the number of ones of a given number in binary.

First adding the number to register $2 and the shift amount in $1 addi $2,$0,31 addi $1, $1, 1

then compare the value to less than zero in register $5 and start the loop label here slt $5, $0, $2

then set the loop condi. to exit when the number equals 0 beq $5, $0 ,7

then see if the first bit is 1 or not by anding it with 1

andi $3,$2,0x01

and then add the result of anding to register $7

add $7,$3,$7

finally shift right logical of the number and jump back to loop srl $2,$2,1

j loop

inputs:31

expected output:5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **instruction** | | **Binary** | **Hexadecimal** |
|  | **1.** | addi $2,$0,31 | 1000 000 010 011111 | 809F |
|  | **2.** | addi $1, $1, 1 |  | 8241 |
|  |  | loop |  |  |
|  | **3.** | slt $5, $0, $2 | 0001 000 010 101010 | 10AA |
|  | **4.** | beq $5, $0 , 8 | 0100 101 000 001000 | 4A08 |
|  | **5.** | andi $3,$2,0x01 | 0110 010 011 000001 | 64C1 |
|  | **6.** | add $7,$3,$7 | 0001 011 111 111000 | 17F8 |
|  | **7.** | srl $2,$2,1 | 0000 010 001 010101 | 0455 |
|  | **8.** | j loop | 1001 000 000 000010 | 9002 |

## b. Selection sort procedure:

The purpose of this program is to sort an array of 8 words read from the memory.

inputs: 8 words from the memory expected output: sorted the 8 words in ascending order.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Instruction** | **Bin Hex** | |
| **1** | addi $2, $2, 7 | 1000 010 010 000 111 | 8487 |
| **2** | add $7, $2, $0 | 0001 010 000 111 000 | 1438 |
|  | Outer Loop |  |  |
| **3** | xor $1, $1, $1 | 0000 001 001 001 011 | 024b |
| **4** | beq $2, $1, 31 | 0100 010 001 011 111 | 445F |
| **5** | add $6, $0, $2 | 0001 000 010 110 000 | 10b0 |
|  | Inner Loop |  |  |
| **6** | beq $1, $2, 10 | 0100 001 010 001 010 | 428a |
| **7** | lw $7, 0($6) | 0010 110 111 000 000 | 2dc0 |
| **8** | lw $3, 0($1) | 0010 001 011 000 000 | 22c0 |
| **9** | slt $4, $3, $7 | 0001 011 111 100 010 | 17e2 |
| **10** | blez $4, 3 | 1101 100 000 000 011 | d803 |
| **11** | addi $1, $1, 1 | 1000 001 001 000 001 | 8241 |
| **12** | j 5 | 1001 000 000 000 110 | 9005 |
|  | Update Max |  |  |
| **13** | add $6, $0, $1 | 0001 000 001 110 000 | 1070 |
| **14** | addi $1, $1, 1 | 1000 001 001 000 001 | 8241 |
| **15** | j 5 | 1001 000 000 000 110 | 9005 |
|  | Swap |  |  |
| **16** | lw $3, 0($2) | 0010 010 011 000 000 | 24c0 |
| **17** | lw $7, 0($6) | 0010 110 111 000 000 | 2dc0 |
| **18** | sw $7, 0($2) | 0011 010 111 000 000 | 35c0 |
| **19** | sw $3, 0($6) | 0011 110 011 000 000 | 3cc0 |
| **20** | addi $2, $2, -1 | 1000 010 010 111 111 | 84bf |
| **21** | j 2 | 1001 000 000 000 011 | 9002 |

## c. Remaining untested instructions:

1. AND, OR, NOR

Purpose is to test and, or, nor with the numbers 5 and 6 Input: 5 into register 2, and 6 into register 3 Expected output:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Instruction** | **Bin** | **Hex** |
| **1** | addi $2, $2, 5 | 1000 010 010 000 101 | 8485 |
| **2** | addi $3, $3, 6 | 1000 011 011 000 110 | 86C6 |
| **3** | and $1, $2, $3 | 0000 010 011 001 000 | 04C8 |
| **4** | or $4, $2, $3 | 0000 010 011 100 001 | 04E1 |
| **5** | nor $5, $2, $3 | 0000 010 011 101 010 | 04EA |

1. SLL, SRA, ROL

Purpose to test sll, sra, rol with the numbers 5 and 6

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **instruction** | **bin** | **hex** |
| **1** | addi $1, $1, 1 | 1000 001 001 000 001 | 8241 |
| **2** | addi $2, $2, 5 | 1000 010 010 000 101 | 8485 |
| **3** | sll $3, $2, $1 | 0000 010 001 011 100 | 045C |
| **4** | srl $4, $2, $1 | 0000 010 001 100 101 | 0465 |
| **5** | sra $5, $2, $1 | 0000 010 001 101 110 | 046E |
| **6** | rol $6, $2, $1 | 0000 010 001 110 111 | 0477 |

1. SUB, SLTU, SLTE, SLTEU, JR

Purpose is to test sub, sltu, slte, slteu, and jr with the numbers 5 and 6

Input: 5 in register 2, and 6 in register 3 Expected output:

Result of sub = 1 in register 5

Result of slt = 1 in register 6

Result of sltu = 1 in register 7 Result of slte = 1 in register 4 jr will jump to address 1

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Instruction** | **bin** | **hex** |
| **1** | addi $1, $1, 5 | 1000 001 001 000 101 | 8245 |
| **2** | addi $2, $2, 5 | 1000 010 010 000 101 | 8485 |
| **3** | addi $3, $3, 6 | 1000 011 011 000 110 | 86C6 |
| **4** | sub $5, $3, $2 | 0001 011 010 101 001 | 16A9 |
| **5** | slt $6, $2, $3 | 0001 010 011 110 010 | 14F2 |
| **6** | sltu $7, $2, $3 | 0001 010 011 111 011 | 14FB |
| **7** | slte $4, $2, $1 | 0001 010 001 100 100 | 1464 |
| **8** | JR 1 | 0001 000 000 001 110 | 100E |

1. ORI, BNE, JALR

Purpose is to test ori, bne, jalr with the numbers 5 and 7 and bne wont branch, jalr will jump. Input: 5 in register 2, and 7 in register 3

Expected output: ori 3 and 7 => 7

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **instruction** | **bin** | **hex** |
| **1** | addi $2, $2, 5 | 1000 010 010 000 101 | 8485 |
| **2** | andi $3, $2, 7 | 0110 010 011 000 111 | 64C7 |
| **3** | ori $4, $3, 3 | 0111 011 100 000 011 | 7703 |
| **4** | bne $1, $3, 8 | 0101 001 011 001 000 | 52C8 |
| **5** | jalr $2, $1 | 0001 001 000 010 111 | 1217 |

1. BLTZ, BLEZ

Purpose is to test bltz and blez with the number zero, bltz won’t branch, blez will branch to address 7

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **instruction** | **bin** | **hex** |
| **1** | bltz $0, 7 | 1100 000 000 000 111 | C007 |
| **2** | blez $0, 7 | 1101 000 000 000 111 | D007 |

1. BGTZ, BGEZ

Purpose to test bgtz, and bgez with number zero bgtz won’t branch, bgez will branch to address 7

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **instruction** | **bin** | **hex** |
| **1** | bgtz $0, 7 | 1110 000 000 000 111 | E007 |
| **2** | bgez $0, 7 | 1111 000 000 000 111 | F007 |

1. JAL, LUI

Purpose is to test jal and lui, jal will jump to address 3, and lui assuming it is at address 3 will load upper immediate the value 5 into register 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **instruction** | **bin** | **hex** |
| **1** | jal 3 | 1011 000 000 000 011 | B003 |
| **2** | lui 5 | 1010 000 000 000 101 | A005 |

3.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Task** |  |
| **Week** | **Design and**  **Implementation** | **simulation and testing** | **design and results reporting** |
| **10** | Dividing tasks | - | - |
| **11** | The register file, ALU | - | - |
| **12** | Instruction Memory, Date  Memory | number of 1’s in a register | write the report |
| **13** | PC, Next PC, MC, ALUCtrl,  Datapath  **In short finish up the single processor** | Selection sort procedure, Max procedure, tests each of the remaining untested instructions | Testing’s, write the report. |
| **14** | Pipelined processor | - | write the report |
| **15** | Pipelined processor  Implementing forwarding, and stalls.  **In short finish up the pipeline processor** | Testing the pipeline correctness | Finish the report |

### 