**Milestone 3: I-Type and J-Type Instructions**

**20% of Project**

**Due: Tuesday, November 21**

In this milestone, your pipeline processor need to successfully run the required I-Type and J-Type MIPS instructions required in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| I-Type Instruction | | | | | |
| Instruction | OPCode | RS | RT | Immediate |
| **andi** $2, $at, 20 | 001100 | 00001 | 00010 | 0000000000010100 |
| **ori** $2, $at, 20 | 001101 | 00001 | 00010 | 0000000000010100 |
| **slti** $2, $at, 20 | 001010 | 00001 | 00010 | 0000000000010100 |
| **addi** $2, $at, 20 | 001000 | 00001 | 00010 | 0000000000010100 |
| **addiu** $2, $at, 20 | 001001 | 00001 | 00010 | 0000000000010100 |
| **beq** $at, $2, 20 | 000100 | 00001 | 00010 | 0000000000010100 |
| **bne** $at, $2, 20 | 000101 | 00001 | 00010 | 0000000000010100 |
| **bgtz** $at, 20 | 000111 | 00001 | 00000 | 0000000000010100 |
| **bgez** $at, 20 | 000001 | 00001 | 00000 | 0000000000010100 |
| **lw** $at, 20($2) | 100011 | 00010 | 00001 | 0000000000010100 |
| **sw** $at, 20($2) | 101011 | 00010 | 00001 | 0000000000010100 |
| **lui** $at, 20 | 001111 | 00000 | 00001 | 0000000000010100 |

|  |  |  |
| --- | --- | --- |
| J-Type Instruction | | |
| Instruction | OPCode | Address |
| **j L5** | 000010 | 00000100000000000000001000 |
| **jal L5** | 000011 | 00000100000000000000001000 |

Note:

* ***bgtz rs, #offset***. Branch on Greater Than Zero.
* ***bne rs, rt, #offset***, Branch on Not Equal
* ***beq rs, rt, #offset*** Branch on Equal
* ***lui rt, #imm***, Load upper immediate. The immediate value is shifted left 16 bits and stored in the register. The lower 16 bits are zeroes.
* ***j target***. **(jump)** Jump to the effective target address.
* ***jal target***. **(jump and link)** Jump to the effective target address. **jal** is commonly used for function calls. **jal** should really be called **laj** for “link and jump”:
  + Step 1 (link): Save address of next instruction into $ra
  + Step 2 (jump): Jump to the given label

# Test program for arithmetic (without read after write hazards)

|  |
| --- |
| .text  main:   1. ori $1, $0, 0xFF ; $1 = 0x0000,00FF 2. ori $2, $0, 0x3F ; $2 = 0x0000,003F 3. andi $2, $1, 0x14 ; $2 = 0x0000,0014 4. ori $2, $1, 0x14 ; $2 = 0x0000,00FF 5. slti $2, $1, 0x14 ; $2 = 0x0000,0000 6. slti $2, $1, 0xFFF ; $2 = 0x0000,0001 7. addi $2, $1, 0x14 ; $2 = 0x0000,0113 8. addiu $2, $1, 0x14 ; $2 = 0x0000,0113 9. lui $1, 0xEE ; $1 = 0x00EE,0000 10. nop 11. nop 12. nop 13. nop |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instruction | Address | Code | Results |
| **1** | ori $1, $0, 0xFF | 0x00400000 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | ori $2, $0, 0x3F | 0x00400004 | **3402003f** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | andi $2, $1, 0x14 | 0x00400008 | **30220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **4** | ori $2, $1, 0x14 | 0x0040000c | **34220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | slti $2, $1, 0x14 | 0x00400010 | **28220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **6** | slti $2, $1, 0xFFF | 0x00400014 | **28220fff** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **7** | addi $2, $1, 0x14 | 0x00400018 | **20220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **8** | addiu $2,$1,0x14 | 0x0040001C | **24220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **9** | lui $1, 0xEE | 0x00400020 | **3c0100ee** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | nop | 0x00400024 | **00000000** |  |
|  | nop | 0x00400024 | **00000000** |  |
|  | nop | 0x00400024 | **00000000** |  |
|  | nop | 0x00400024 | **00000000** |  |

# Test program for arithmetic (With read after write hazards)

|  |
| --- |
| .text  main:   1. ori $2, $0, 0x3F ; $2 = 0x0000,003F 2. ori $1, $0, 0xFF ; $1 = 0x0000,00FF 3. andi $2, $2, 0x14 ; $2 = 0x0000,0014 4. ori $2, $2, 0x15 ; $2 = 0x0000,0015 5. slti $2, $2, 0x14 ; $2 = 0x0000,0000 6. slti $2, $2, 0xFFF ; $2 = 0x0000,0001 7. addi $2, $2, 0x14 ; $2 = 0x0000,0015 8. ori $1, $0, 0xFF ; $1 = 0x0000,00FF 9. ori $1, $0, 0xFF ; $1 = 0x0000,00FF 10. addiu $2, $2, 0x14 ; $2 = 0x0000,0029 11. lui $2, 0xEE ; $2 = 0x00EE,0000 12. nop 13. nop 14. nop 15. nop |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instruction | Address | Code | Results |
| **1** | ori $2, $0, 0x3F | 0x00400000 | **3402003f** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | ori $1, $0, 0xFF | 0x00400004 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | andi $2, $2, 0x14 | 0x00400008 | **30420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **4** | ori $2, $2, 0x15 | 0x0040000C | **34420015** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | slti $2, $2, 0x14 | 0x00400010 | **28420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **6** | slti $2, $2, 0xFFF | 0x00400014 | **28420fff** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **7** | addi $2, $2, 0x14 | 0x00400018 | **20420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **8** | ori $1, $0, 0xFF | 0x0040001C | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **9** | ori $1, $0, 0xFF | 0x00400020 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **10** | addiu $2,$2, 0x14 | 0x00400024 | **24420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **11** | lui $2, 0xEE | 0x00400028 | **3c0200ee** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | nop | 0x0040002C | **00000000** |  |
|  | nop | 0x0040002C | **00000000** |  |
|  | nop | 0x0040002C | **00000000** |  |
|  | nop | 0x0040002C | **00000000** |  |

# Test programs for branch on equal (beq)

|  |  |  |
| --- | --- | --- |
|  | Test case: Branch is taken | Test case: Branch is NOT taken |
| Test Program | **.text**  **main:**  **ori $1, $0, 1**  **ori $2, $0, 1**  **beq $1, $2, L**  **nop**  **ori $4, $0, -1**  **L: ori $3, $0, 1** | **.text**  **main:**  **ori $1, $0, 2**  **ori $2, $0, 1**  **beq $1, $2, L**  **nop**  **ori $4, $0, -1**  **L: ori $3, $0, 1** |
| If it runs correctly | $4=0, $3=1 | $4=-1, $3=1 |
| If it runs incorrectly | $4=-1, $3=1 | $4=0, $3=1 |

# Test programs for branch on not equal (bne)

|  |  |  |
| --- | --- | --- |
|  | Test case: Branch is taken | Test case: Branch is NOT taken |
| Test Program | **.text**  **main:**  **ori $1, $0, 2**  **ori $2, $0, 1**  **bne $1, $2, L**  **nop**  **ori $4, $0, -1**  **L: ori $3, $0, 1** | **.text**  **main:**  **ori $1, $0, 1**  **ori $2, $0, 1**  **bne $1, $2, L**  **nop**  **ori $4, $0, -1**  **L: ori $3, $0, 1** |
| If it runs correctly | $4 = 0, $3 = 1 | $4 = -1, $3 = 1 |
| If it runs incorrectly | $4 =-1, $3 = 1 | $4 = 0, $3 = 1 |

# Test program for jump

|  |
| --- |
| **.text**  **main:**  **ori $1, $0, 1**  **j, L**  **nop**  **ori $4,$0, -1**  **L: ori $3, $0, 1** |

If the codes run correctly, then $4=0 and $3=1. If $4=-1, then your project does not jump correctly.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Instruction | Address | | Code | | Results | |
| **1** | **ori** $1, $0, 1 | | 0x00400000 | | **0x34010001** | | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | **j** L | | 0x00400004 | | **0x08100004** | | PC = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | **nop** | | 0x00400005 | | **0x00000000** | |  |
| **4** | **ori** $4, $0, -1 | | 0x0040000c | | **0x3404ffff** | | $4 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | **L: ori** $3, $0, 1 | | 0x00400010 | | **0x34030001** | | $3 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# Test program for jump-and-link

|  |
| --- |
| **.text**  **main:**  **ori $1, $0, 1**  **jal, L**  **nop**  **ori $4, $0, -1**  **L: ori $3, $0, 1** |

The ***jal*** instruction saves the return address in register $31. This register is also called $ra (where "***ra***" means return address).

If the codes run correctly, then $4=0 and $3=1. If $4=-1, then your project does not jump correctly. In addition, *$31 should be correctly set*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instruction | Address | Code | Value of $4, $3, and $31 |
| **1** | **ori** $1, $0, 1 | 0x00400000 | **0x34010001** | $3 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | **jal** L | 0x00400004 | **0x0c100006** | PC = \_\_\_\_\_\_\_\_\_\_\_\_\_\_  $31 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | **nop** | 0x00400005 | **0x00000000** |  |
| **4** | **ori** $4, $0, -1 | 0x0040000c | **0x3404ffff** | $4 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | **L: ori** $3, $0, 1 | 0x00400010 | **0x34030001** | $3 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# Comprehensive test program

##############

# Register map

##############

# $s0 --> Sum of the array

# $t0 --> Address of the integer array

##############

.data # following words in data segment

A: .word 1 2 3 4 5 6 7 8 9 10 0 # array of 10 words in location A

.globl main # define global label main

.text # text segment begins here

main:

la $t0, A # load address of A in $t0

add $s0, $zero, $zero # s0 = 0

loop:

lw $t2, 0($t0) # $t2 = Mem[$t0 + 0]

add $s0, $s0, $t2 # $s0 = $s0 + $t2

addi $t0, $t0, 4 # $t0 = $t0 + 4

bne $t2, $zero, loop # if ($t2 != 0) goto loop

nop # else do nothing

exit: beq $0, $0, exit # program stops here

nop

Note that la is a pseudo-instruction, which is not really a MIPS instruction but it is allowed in assembly language code. The assembler translates a pseudo instruction into MIPS code. This makes the job of writing this assembly language code easier. For example,

***la $t0, array***

where $t0 is pointer into an array of integers. If this integer array is assigned address 0x00aa0bb0, then the assembler will assign the value 0x00aa0bb0 to $t0. However, while the memory address is 32 bits, one machine instruction is only 32 bits wide. Accordingly this assignment can be implemented by using one instruction. As a result, two instructions are required to implement this instruction.

lui $t0, 0x00aa # $t0 gets value 0x 00aa 0000

ori $t0, $t0, 0x0bb0 # $t0 gets value 0x 00aa 0bb0

**Milestone 3: I-Type and J-Type Instructions**

**20% of Project Grade**

**TA Checkoff Sheet**

Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Score: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TA Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Test program for Arithmetic Instruction without hazards (18 points)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instruction | Address | Code | Results |
| **1** | ori $1, $0, 0xFF | 0x00400000 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | ori $2, $0, 0x3F | 0x00400004 | **3402003f** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | andi $2, $1, 0x14 | 0x00400008 | **30220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **4** | ori $2, $1, 0x14 | 0x0040000c | **34220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | slti $2, $1, 0x14 | 0x00400010 | **28220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **6** | slti $2, $1, 0xFFF | 0x00400014 | **28220fff** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **7** | addi $2, $1, 0x14 | 0x00400018 | **20220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **8** | addiu $2,$1,0x14 | 0x0040001C | **24220014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **9** | lui $1, 0xEE | 0x00400020 | **3c0100ee** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | nop | 0x00400024 | **00000000** |  |

# Test program for Arithmetic Instruction with hazards (22 points)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instruction | Address | Code | Results |
| **1** | ori $2, $0, 0x3F | 0x00400000 | **3402003f** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **2** | ori $1, $0, 0xFF | 0x00400004 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **3** | andi $2, $2, 0x14 | 0x00400008 | **30420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **4** | ori $2, $2, 0x15 | 0x0040000C | **34420015** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **5** | slti $2, $2, 0x14 | 0x00400010 | **28420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **6** | slti $2, $2, 0xFFF | 0x00400014 | **28420fff** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **7** | addi $2, $2, 0x14 | 0x00400018 | **20420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **8** | ori $1, $0, 0xFF | 0x0040001C | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **9** | ori $1, $0, 0xFF | 0x00400020 | **340100ff** | $1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **10** | addiu $2,$2,0x14 | 0x00400024 | **24420014** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **11** | lui $2, 0xEE | 0x00400028 | **3c0200ee** | $2 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | nop | 0x0040002C | **00000000** |  |

# Branch Instructions (40 points)

|  |  |  |
| --- | --- | --- |
| Branch Instruction | Test Case | Pass or Fail |
| beq | Branch Taken (5 points) |  |
| Branch Not Taken (5 points) |  |
| bne | Branch Taken (5 points) |  |
| Branch Not Taken(5 points) |  |
| j | (10 points) |  |
| jal | (10 points) |  |

# Test Program: Calculate the sum of an array (20 points)

If your program runs correctly, you get the full credit. However, if not, you can earn the following partial credits.

* Hand the stall required for the read after load. (3 points)
* LW works correctly. (5 points)
* The loop body (calculating the sum) is executed incorrectly. (5 points)
* The dead loop is executed correctly ( 3 points)