Michael Lankford

Homework 3

2/20/2023

1. exercise 2.9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Type | Opcode (op) | Source register 1 (rs) | Source Register 2 (rt) | Destination register (rd) | Immediate field (immed) |
| addi $t0, $s6, 4 | I | 8 | 22 | 8 | - | 4 |
| add $t1, $s6, $0 | R | 0 | 22 | 0 | 9 | - |
| sw $t1, 0($t0) | I | 43 | 8 | 9 | - | 0 |
| lw $t0, 0($t0) | I | 35 | 8 | 8 | - | 0 |
| add $s0, $t1, $t0 | R | 0 | 9 | 8 | 16 | - |

1. exercise 2.11 231 - 1 is the highest MIPS can go and -231 is the lowest
   1. 2.11.1

$t0 = 128 + $s1

$t0 > 231 - 1

128 + $s1 > 231 - 1

If $s1 > 231 - 129 then there will be overflow

$t0 = 128 + $s1

$t0 < -231

128 + $s1 < -231

If $s1 < -231 - 128 then there will be overflow

* 1. 2.11.2

$t0 = 128 - $s1

$t0 > -231 - 1

128 - $s1 > -231 - 1

If $s1 < -231 + 129 then there will be overflow

$t0 = 128 - $s1

$t0 < 231

128 - $s1 < 231

If $s1 > 231 + 128 then there will be overflow

* 1. 2.11.3

$t0 = $s1 – 128

$t0 < -231

$s1 - 128 > 231 - 1

If $s1 > 231 + 127 then there will be overflow

$t0 = $s1 - 128

$t0 < -231

$s1 - 128 < -231

If $s1 < -231 + 128 then there will be overflow

1. exercise 2.13

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Type | Opcode (op) | Source register 1 (rs) | Source Register 2 (rt) | Destination register (rd) | Immediate field (immed) |
| sw $t1, 32($t2) | I | 43 | 10 | 9 | - | 32 |

Op code = 101011 Source 1 = 01010

Source 2 = 01001 Destination = 00000

Shamt = 00000 Funct = 100000

0b10101101010010010000000000100000

The hex representation is: 0xAD490020.

1. exercise 2.14

The type of the instruction is R-type.

The assembly language instruction is sub $v1, $v1, $v0.

Op code = 000000 Source 1 = 00011

Source 2 = 00010 Destination = 00011

Shamt = 00000 Funct = 100010

0b00000000011000100001100000100010

The binary representation is: 0x00621822.

1. exercise 2.21

slt $t2, $0, $t0 🡪 t2 = 1 because 0 < 0x010100000

bne $t2, $0, ELSE 🡪 t2 != register 0 so jump to else

j DONE

ELSE: addi $t2, $t2, 2 🡪 t2 = 1 + 2 = 3

DONE:

$t2 is equal to 3 after these instructions are executed.

1. exercise 2.24
   1. 2.24.1

LOOP: slt $t2, $0, $t1

beq $t1, $t1, DONE

subi $t1, $t1, 1

addi $s2, $s2, 2

j LOOP

DONE:

$s0 is never manipulated so it will remain 0.

* 1. 2.24.2

I = 10;

while(0 < I){

B = B + 2;

I = I - 1;

}

* 1. 2.24.3

Because there are 5 instructions inside the loop and the loop is dependent on the value of the $t1 register, there would be 5N instructions executed.

1. exercise 2.35 0x11223344
   1. 2.35.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Big endian | 11 | 22 | 33 | 44 |

The value stored at 0x10000004 would be 11.

* 1. 2.35.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Little endian | 44 | 33 | 22 | 11 |

The value stored at 0x10000004 would be 44.

1. exercise 2.40 CPI = (CPI \* Execution time)
   1. 2.40.1

CPI = (.7 \* 2) + (.1\* 6) + (.2 \* 3) = 2.6

* 1. 2.40.2

2.6 \* .75 = 1.95 for a 25% improvement

(.7 \* x) + (.1 \* 6) + (.2 \* 3) = .7x + 1.2

1.95 – 1.2 = .7x

.75 = .7x

X = 1.07 cycles on average for arithmetic instructions

* 1. 2.40.3

2.6 \* .50 = 1.3 for a 50% improvement

(.7 \* x) + (.1 \* 6) + (.2 \* 3) =.7x + 1.2

1.3 – 1.2 = .7x

.1 = .7x

X = .14 cycles on average for arithmetic instructions