Homework 3

Tuesday, July 26, 2022 4:41 PM

Q1

addi $t0, $t1, -10 → I-type

op 6 bits	rs 5 bits	rt 5 bits	imm 16 bits
↱8	↱9	↱8	↱ -10
001000	01001	01000	1111111111110110

2 1 2 8 F F F 6

0x2128FFF6

Explanation:
→ I type so use this template
→ use tables to find each value
→ convert to binary
→ conver to hex

#bits 6 5 16

lw $s0, -3($t0) (op, rt, imm, rs) — found on table

I-type so use

100011 01 000 1 0000 1111111111111101) convert to hex

8 D 1 0 F F F D

0X8D10FFFD

beq $t0, $s0, L2 (op, rs, rt, imm) → offset = 1 — on table

0001 0001 0001 0000 0000000000000001

1 1 1 0 0 0 0 1

0x11100001

conditional I-type
label offset is imm value, 16 bits, convert to hex

sw $v0, 84($0) (op, rt, imm, rs)

1010 1100 0000 0010 0000 0000 0101 0100

A C 0 2 0 0 5 4

0x AC020054

I type, use template & tables, follow same explanation as above

addi $t0, $sp, -4 (op, rt, rs, imm)

0010 0011 0110 1000 1111111111111100

2 3 6 8 F F F C

0x2368FFFC

I-type, use template and tables for register values, convert to hex. last # is 16 bits.

beq $t0, $s0, L1 (op, rs, rt, imm) → offset = -4

000100 01 0001 0000 1111111111111100

1 1 1 0 F F F C

0x1110FFFC

Conditional I-type
label offset is immediate value, 16 bits convert to hex

Q2

a) 000000 10010 10001 10010 00000 100000 (op, rs, rt, rd, shamt, funct)
R-type, 18, 17, 18, 0, add

rs = $s2
rt = $s1
rd = $s2
} assembly code: **add $s2, $s2, $s1**

b) 000000 00000 10010 10010 00010 000000 (op, rs, rt, rd, shamt, funct)
R type, 0, 18, 18, 2, 0

rs = $0
rt = $s2
rd = $s2
} assembly code: **sll $s2, $s2, 2**

c) 101011 10010 01110 0000000000000000 (op, rs, rt, imm)
43 = sw, I-type, 18, 14, 0

rs = $s2
rt = $t6
} assembly code: **sw $t6, 0($s2)**

d) 001000 10000 10000 0000000000000001 (op, rs, rt, imm)
8 = addi, 16, 16

rs = $s0
rt = $s0
} assembly code: **addi $s0, $s0, 1**

e) 000101 10000 01000 1111111111110011 (op, rs, rt)
bne, I type, 16, 8, label (-13)

rs = $s0
rt = $t0
} assembly code: **bne $s0, $t0, (-13)** → label offset = -13 (FFF3)

f) 100011 10010 01010 0000000000000101 (op, rs, rt)
35 = lw, I-type, 18, 10, 5

rs = $s2
rt = $t2
} assembly code: **lw $t2, 5($s2)**

Q3

C code
```
sum = 0;
for (i = 0; i<10; i++) {
    sum = sum + L[i];}
```

Mips
```
# $s1 = i, $s2 = base address array L = 0x55552222
        lui   $s2, 0x5555          // load upper immediate
        ori   $s2, $s2, 0x2222
        addi  $s1, $0, 0
        addi  $t2, $0, 10
target: slt   $t0, $s1, $t2
        beq   $t0, $0, done
        sll   $t3, $s1, 2          // t3 = i * 4 (byte offset)
        add   $t3, $t3, $s0        // address of L[i]
        lw    $t1, 0($t3)
```

Q4

$t0 = c, $s0 = x, $s1 = y, $s2 = z

```
switch (c) {
    case 1: x = y + z;
        break;
    case 2: x = y - z;
        break;
    default: x++;
        break;
```

```
        beq $t0, $s0, case0   // case 0
        lw  $t1, 1            // load 1 to compare
        beq $t0, $t1, case1   // case 1
        add $s0, $0, 1        // default
case0:  add $s0, $s1, $s2
case1:  sub $s0, $s1, $s2
```

Q5

$t0 = random 32 bit int. $s0 = store count

```
(initializing)
        lui   $t0, x          } loading the 32 bit
        ori   $t0, $t0, y       int xy to reg $t0.
        addi  $s0, $0, 0      } counter = 0
label 1:
        beq   $t0, 0, label 2
        and   $s1, $t0, 1
        beq   $s1, 0, label2
        add   $s0, $s0, 1
label 2:
        srl   $t0, $t0, 1
        j     label 1
end: ....
```

Q6

a) range is [0, 8) or (0 ≤ $s1 ≤ 7)
every shift left multiplies by 2

if ($s1 < 0) $t1 = 1
else $t1 = 0
so if $t1 != 0 → done — So we want $t1 to equal 0. for $t1 = 0, $s1 must be greater than or equal to 0.
If $t1 == 0
if ($s1 < 8) $t2 = 1
else $t2 = 0
so if $t2 == 0 → done — so we want $t2 = 1 for $t2 = 1, $s1 must be less than 8
else add 1 + $s1
done...

b) because its unsigned so there are no negative numbers. no negative # means must be above and 0.

line1: if ($s1 < 8) $t1 = 1
else $t1 = 0
line2: if $t1 = 0 → done
else, continue
} ∴ we want $t1 in the range [0, 8)

for example: if $s1 = 3

mips
```
sltiu  $t1, $s1, 8
beq    $t1, $0, done
addi   $s1, $s1, 1
```

meaning
if (3 < 8) $t1 = 1
if $t1 == 0 → done ✗
else, continue ✓
increment $s1 by 1
if (4 < 8) $t1 = 1
& repeat until $s1 = 8, then $t1 == 0 → done