Jordan Winkler
Computer Architecture
Mon Feb 25 21:58:45 EST 2019
hw5

1. For the following C code segment, write a code segment in MIPS assembly language to do the same thing. Assume i is in $s0, x is in $s1, and y is in $s2. Dont forget to comment your code. Make sure that you use the slt statement.

```
for (i = 0; i < x; i=i+1)
    y = y + i;
```

```
# translate some c code to mips
# for (i = 0; i < x; i=i+1)
#    y = y + i;
# $s0 = i, $s1 = x, $s2 = y
# required: use slt
for:                            # for (i = 0; i < x; i=i+1) y = y + i;
    add     $s0, $0, $0         # i=0
for_loop:
    slt     $t0, $s0, $s1       # (i < x) ? 1 : 0
    beq     $t0, $0, endfor     # exit if (i < x)
    add     $s2, $s2, $s0       # y = y + i
    addi    $s0, $s0, 1         # i = i + 1
    j       for_loop
endfor:
```

2. Show how the value 0xabcdef12 would be arranged in memory of a little-endian and a big-endian machine. Assume the data is stored starting at address 0.

```
0xabcdef12
big endian
0 ab
1 cd
2 ef
3 12

little endian
0 12
1 ef
2 cd
3 ab
```

3. Translate the following MIPS code to C. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

```
# s0 = f, s1 = g, s2 = h, s3 = i, s4 = j, s6 = A, s7 = B
    addi    $t0, $s6, 4          # t0 = A + 4;
    add     $t1, $s6, $0         # t1 = A;
    sw      $t1, 0($t0)          # t0 = *t1
    lw      $t0, 0($t0)          # t0 = *t0
    add     $s0, $t1, $t0        # f = t1 + t0

# single line C code would be
# f = A + *(*A);
```

4. Find the shortest sequence of MIPS instructions that extracts the 18-bit field in locations 5 through 22 (bit position 31 is the MSB) from register $t3 and places it in register $t0. Hint: Use shift instructions.

```
    # extract 5-22 bit postion from t3
    sll     $t0, $t3, 10 # take off top,    32 = 22+10
    srl     $t0, $t0, 15 # take off bottom, -5 = 10-15
```

5. Assemble the following MIPS instruction: srl $s1, $t2, 3

```
srl, shift_right_logical, R-type, R[rd] = R[rt] >> shamt
srl $s1, $t2, 3
000000 00000 t2 s1 3 srl
000000 00000 10d 17d 00011 000010
000000 00000 01010 10001 00011 000010
```

6. Convert the following C/C++ fragment into equivalent MIPS assembly language. Assume that the variables a, b, c, d, i and x are assigned to registers $t1, $t2, $t3, $t4, $s0 and $s1 respectively.

```
if ((a < b) && (c == 0))
  d = 1;

# if ((a < b) && (c == 0))
#   d = 1;
# a,  b,  c,  d,  i,  x
# t1, t2, t3, t4, s0, s1
```

```
    slt     $t0, $t1, $t2       # a <  b
    seq     $t8, $t1, $0        # c == 0
    and     $t0, $t0, $t8       # (a <  b) && (c == 0)
    beq     $t0, $0, skip       # if ((a < b) && (c == 0))
    addi    $t4, $0, 1          #   d = 1;
skip:
```