CS-GY 6133: Computer architecture HW1

Qiming Zhang NetID: qz718

Yuzheng Wang NetID: yw2787

September 29, 2016

Total: 115pts

Problem 1

(a)

10pts

Figure 1: instruction format for subleq instruction

Explain: Since our computer is 4GB byte-addressable, the length of each address is 32bits. Since a,b,c are all memories, and there is only one instruction in Turing complete, we don't need any opcode to encode one instruction, we can conclude that the total length of one instruction is 3*32+0 = 96bits.

10pts

(b)

3 address ISA. Memory-memory ISA.

18pts

(c)

```
L0: subleq C, C, L1          #  C=C-C=0<=0 go to L1, C=0
L1: subleq D, D, L2          #  D=D-D=0<=0 go to L2, D=0
L2: subleq A, B, L6          #  A=A-B=16>0 go to L3, A=16
L3: subleq D, B, L4          #  D=D-B=9>0 go to L4, D=9
L4: subleq C, D, L5          #  C=C-D=-9<=0, go to L5, C=-9
L5: subleq D, D, L9          #  D=D-D=0<=0, go to L9, D=0
L6: subleq D, A, L7          #  not executed
L7: subleq C, D, L8          #  not executed
L8: subleq D, D, L9          #  not executed
L9: //Rest of code           #  end
```

Finally, A = 16, B = -9, C = -9, D = 0; this snippet tries to compare the value stored in A and B, if A>B, it will store the smaller value into C and clear the value in D, and it will store the difference between A and B into A. If A<=B, it will store the difference of A-B into A and C, also the value stored in D will be cleared. In two cases, the value stored in B remains constant.

20pts

(d)

```
L1: Subleq M2, M2, L2        # clear M2 to 0
L2: Subleq M2, M0, L3        # M2 = M2 - M0 = -N
L3: Subleq M1, M2, L4        # M1 = M1 - M2 = N, we assume M1
                             equals to 0 initially
```

```
L4: Subleq M0, C1, Lend          # M0 = M0 - C1 = N -1
L5: Subleq M2, M2, L2            # clear M2 to 0 and jump back
                                 to L2
Lend:
```

Problem 2

5pts

(a)

Didn't consider the corner cases such as overflow. 5pts

```
Sub rx, rx, rx                   # clear rx
Sub rx, rt, rx                   # rx = rx - rt = - rt
Sub rs, rx, rd                   # rd = rs - rx = rs + rt
```

(b)

5pts

```
Or rs, rt, rd                    #rd = rs | rs
Xor rs, rt, rt                   # get xor result into rt
Sub rd, rt, rd                   # substract rd with rt
Xor rs, rt, rt                   # recover rt
```

Using sub in logical algebra?

(c)

8pts

```
Lhu rs, rt, imm+2                # get the last two bytes, 0 extension
Sub rx, rx, rx                   # clear rx to 0
Lhu rs, rx, imm                  # get the first two bytes
Sll rt, rt, 16                   # left shift rt
Add rx, rt, rt                   # add two register
```

addu/or

(d)

7pts

```
Add $0, rx, jumpAddress
Jr rx
```

You should also change PC if using Jr -3pts

this one is not necessarily correct, because jumpAddress does not include the 4 MSB of PC. We also try beq rx,rx,jumpAddress, but this also does not work, since PC = PC + jumpAddress, which is not the same with jumpAddress. We cannot think of other correct choices so we just use this as our answer.

Problem 3

10pts

(a)

3-address
Thumb : ADD Rd, Rs, Rn ARM : ADDS Rd, Rs, Rn
2-address
Thumb : CMP Rd, Rs ARM : CMP Rd, Rs

(b)

Displaced:

1	0	LDR Rd, [Rb, #Imm]	LDR Rd, [Rb, #Imm]	Calculate the source address by adding together the value in Rb and Imm. Load Rd from the address.

Indexd :

1	0	LDR Rd, [Rb, Ro]	LDR Rd, [Rb, Ro]	Pre-indexed word load: Calculate the source address by adding together the value in Rb and the value in Ro. Load the contents of the address into Rd.

Absolute: this can be considered as one absolute, since Rd = M(PC+#imm)

0	ADD Rd, PC, #Imm	ADD Rd, R15, #Imm	Add #Imm to the current value of the program counter (PC) and load the result into Rd.
1	ADD Rd, SP, #Imm	ADD Rd, R13, #Imm	Add #Imm to the current value of the stack pointer (SP) and load the result into Rd.

20pts for three modes above and memory indirect

0pts for auto-increment, format 15 multi load/store is a example

2pts for register indirect mode since there is no explanation