Cody Martin

18-447

Homework #1

2. Little Endian: 10100000(byte 0) 11010110(byte 1) 10001111(byte 2) 00110100(byte 3)

Big Endian: 00110100(byte 3) 10001111(byte 2) 11010110(byte 1) 10100000(byte 0)

3. (1) Zero-address machine:

PUSH B

PUSH C

ADD

POP A

PUSH A

PUSH C

ADD

POP B

PUSH A

PUSH B

SUB

POP D

3 + 3 + 1 + 3 + 3 + 3 + 1 + 3 + 3 + 3 + 1 + 3 = 30 bytes of code

4\*(9) = 36 bytes of data transferred

(2) one-address machine

LOAD B

ADD C

STORE A

ADD C

STORE B

LOAD A

SUB B

STORE D

3\*8 = 24 bytes of code

4\*8 = 32 bytes of data transferred

(3) two-address machine

SUB A, A

ADD B, C

ADD A, B

ADD C, A

SUB B, B

ADD B, C

SUB A, B

SUB D, D

ADD D, A

9\*5 = 45 bytes of code

9\*12 = 108 bytes of data transferred

(4a) three-address memory-memory machine

ADD A, B, C

ADD B, A, C

SUB D, A, B

7\*3 = 21 bytes of code

12\*3 = 36 bytes of data transferred

(4b) three-address load-store machine

LD R1, B

LD R2, C

ADD R3, R2, R1

ST R3, A

ADD R1, R2, R3

ST R1, B

SUB R2, R3, R1

ST R2, D

4\*8 = 32 bytes of code

4\*5 = 20 bytes of data transferred

Out of all these machines, the three address memory-memory machine is the most efficient in terms of code size. In terms of total memory bandwidth, the three address load-store machine is the most efficient with 52 bytes of code and data transfers.`

4.1 start: add $8, $4, $0 #n

lui $9, 0 #a

lui $10, 1 #b

lui $11, 1 #c

lui $12, 1 #loop\_breaker

loop: beq $8, $12, cleanup

nop

add $11, $10, $9

add $9, $10, $0

add $10, $11, $0

addiu $8, $8, -1

j loop

cleanup: add $2, $11, $0

jr $31

4.2 a) $8 will be ECX, $9 will be ESI, and $10 will be EDI, $11 is a storing register

loop: blez $8, next\_instruction

lb $11, 0($9)

sb $11, 0($10)

addi $9, $9, 1

addi $10, $10, 1

addiu $8, $8, -1

j loop

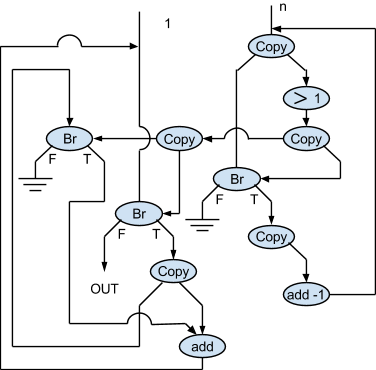
next\_instruction: following instruction

b) The MIPS implementation is 28 bytes, much bigger than the 2 bytes the x86 occupies.

c) Loop will run 0xdeadbeef times, that are 7 instructions long then

blez $8, next\_instruction and lb $11, 0($9) will execute once more. So, 0xdeadbeef\*7 + 2 = 0x616C0388B.

d) Loop will run 0x01000000 times, so 0x01000000\*7 + 2 = 0x07000002 instructions will be run.



5.

6. a) Not necessarily. The instructions being currently processed may vary in size and complexity.

b) No. The slower processor may just have more complex instructions.

7. ISA A: 5 billion instructions per second

ISA B: 1.2 billion instructions per second

We don’t know who has higher performance. It looks like ISA A does, but its instructions may be much more simple than the instructions in ISA B. So even though it processes many more instructions in a second, ISA A’s functionality may not be any better than ISA B’s functionality.