Willis Allstead
4/23/18
CS 219

HW10

12.6

PURPOSE: I am trying to understand how different instruction counts can affect code length required to perform certain tasks, and also they ways those tasks are performed given more or less instructions per line.

X=(A+B×C)/(D−E×F)

0 Address Machine:
```
PUSH A;
PUSH B;
PUSH C;
MUL;
ADD;
PUSH D;
PUSH E;
PUSH F;
MUL;
SUB;
DIV;
POP X;
```
1 Address Machine:
```
LOAD E;
MUL F;
STORE R;
LOAD D;
SUB R;
STORE R;
LOAD B;
MUL C;
ADD A;
DIV R;
STORE X;
```
2 Address Machine:
```
MOVE R1, E;
MUL R1, F;
MOVE R2, D;
SUB R2, R1;
MOVE R1, B;
MUL R1, C;
ADD R1, A;
DIV R1, R2;
MOVE X, R1;
```
3 Address Machine:
```
MUL R1, E, F;
SUB R1, D, R1;
MUL R2, B, C;
ADD R2, A, R2;
DIV X, R1, R2;
```

12.7

PURPOSE: I am trying to understand how to make more complex operations such as data transfer and conditional branches from a very limited number of instructions, in this case two.

a) Data transfer: Location X to accumulator, accumulator to location X.

```
SUBS B;
SUBS B;
SUBS X;
SUBS B
SUBS B;
SUBS X;
SUBS B;
SUBS X;
SUBS X;
SUBS B;
SUBS X;
SUBS B;
SUBS B;
SUBS X; At this point the original value of X is in the
location of X and the accumulator
```

b) Conditional Branch:

```
SUBS A; store accumulator in A
SUBS B; zeroing-out accum. and B
SUBS B;
SUBS BCH;     X and accum. are negative of the first X value
SUBS A;
SUBS BCH;
SUBS BCH;
SUBS A;
SUBS BCH;
SUBS A;
SUBS A; zero out the A
JUMP BCH+1;
```

13.1

PURPOSE: I am trying to understand what different instructions result in with different sized memory and a one-address machine. The different load types are the main

Word 20 contains 40
Word 30 contains 50
Word 40 contains 60
Word 50 contains 70

LOAD IMMEDIATE 20: Data = 20
LOAD DIRECT 20: Data = 40
LOAD INDIRECT 20: Data = 60
LOAD IMMEDIATE 30: Data = 30
LOAD DIRECT 30: Data = 50
LOAD INDIRECT 30: Data = 70

13.3

PURPOSE: I am trying to learn how using different addressing modes affects the corresponding operand locations.

immediate addressing?

Address of operand = 14

direct addressing?

Address of operand = Location of 14 in memory

indirect addressing?

Address of operand = Memory location with address in the location of 14 in memory

register addressing?

Address of operand = Register 14

register indirect addressing?

Address of operand = Memory location with address stored in register 14

13.5

PURPOSE: I am trying to learn how branch instruction lengths and signed displacements affect the branch target address in a PC.

256028 + 3 = 256031 because instruction length is 3 bytes.
256031 – 31 = 256000 with signed displacement
Branch target address = 256000

13.7

PURPOSE: I am trying to learn the number of times the processor needs to refer to memory when it executes in an indirect-addressing mode given different instruction types.

a) A Computation requiring a single operand
1 to fetch the instruction, 1 to fetch the effective address or operand address, 1 to fetch the operand.
In total -> 3 times
b) A branch
1 to fetch the instruction, 1 to fetch the effective address or operand address and transfer to program counter
In total -> 2 times

13.11

PURPOSE: I am trying to learn how to find the address of the operand given a processor that has a displacement, a base, and an index.

Displacement = 1970 decimal
Base = 48022
Index = 8
Address of the operand = 48,022 + 8 + 1,970 = 50,000