1. ~~]`a)~~

~~i) Load MemSrc RegDst~~

~~6 bit Opcode for sending control signals to CPU,~~

~~8 bit MemAddress points to where data is loaded from,~~

~~8 bit register address as the destination address to the register where data is~~

~~stored into.~~

~~Add RegA RegB RegDst~~

~~6 bit Opcode for sending control signals to CPU,~~

~~8 bit RegA points to data a,~~

~~8 bit RegB points to data b,~~

~~8 bit RegDst points to register where the result of (a+b) is stored~~

~~Shl RegSrc RegDst~~

~~6 bit Opcode for sending control signals to CPU,~~

~~8 bit RegSrc points to data a,~~

~~8 bit RegDst points to register where the result of (a << 1) is stored~~

load RegDst MemSrc

6 bit opcode

5 bit regDst

21 bit MemSrc

0 bit unused

add RegA RegB RegDst (OR add RegDst RegA RegB)

6 bit opcode

5 bit RegA

5 bit RegB

5 bit RegDst

11 bit unused

shl RegSrc RegDst ShiftValue (OR shl RegDst RegSrc ShiftValue)

6 bit opcode

5 bit RegSrc points to data a,

5 bit RegDst points to register

5 bit ShiftValue (32 bit processor only needs to shift a value by a maximum of 31 bits)

11 bit unused

Surely if there are 32 registers we can index through in 5 bits (2^5 = 32)

ii)

* + Accessing main memory may take multiple cycles (load).
  + Commands which work on two register addresses (add) will have to take a separate cycle to load each, as opposed to one that only works on one (shl).

b)

Assume input and output at register 0:

shl 0 1 2 # 0=x, 1=4\*x

add 0 1 0 # 0=5\*x

3 Cycles

For k = 508:

shl 0 0 2 # 0=4\*x

shl 0 1 1 # 0=4\*x, 1=8\*x

add 0 1 1 # 0=4\*x, 1=12\*x

shl 1 1 1 # 0=4\*x, 1=24\*x

add 0 1 1 # 0=4\*x, 1=28\*x

shl 1 2 1 # 0=4\*x, 1=28\*x, 2=56\*x

add 0 2 2 # 0=4\*x, 1=28\*x, 2=60\*x

shl 2 2 3 # 0=4\*x, 1=28\*x, 2=480\*x

add 1 2 0 # 0=508\*x

13 Cycles (There may be an improvement on this)

c)

1. P1 is 70% quicker than P2
2. P2 is 23% (2sf) quicker than P1

2a)

loop (int a, int b, int c) {

int x, y;

y = b;

for (x = a + c; x > 0; x--) {

y++;

return y;

}