Unassessed Course Work 3: Answers

1.

```
n = 32
addr = 1F0H
loop exit when n <= 0
        if Memory[addr] < 0 then
                Memory[addr] = - Memory[addr]
        end if
        addr = addr+1
        n = n - 1
end loop
```

Register Allocation. We'll use:

Register 1 for *n*
Register 2 for *addr*.
Register 3 will be employed as a temporary register.

Addr	Assembler Instr	Comment	Machine Instruction
0	0	Zero	0000 0000 0000 0000
1	1	One	0000 0000 0000 0001
2	32	No of integers	0000 0000 0010 0000
3	1F0H	Start address	0000 0001 1111 0000
....			
10H	**LOAD** R1, [2H]	n = 32	0001 0100 0000 0010
11H	**LOAD** R2, [3H]	addr =1F0H	0001 1000 0000 0011
12H	**IFZER** R1, 1DH	**loop exit when** n<=0	0110 0100 0001 1101
13H	**IFNEG** R1, 1DH		0111 0100 0001 1101
14H	**LOAD** R3, [R2]	R3 =Memory[addr]	1001 1110 xxxx xxxx
15H	**IFNEG** R3, 17H	**if** R3 < 0	0111 1100 0001 0111
16H	**GOTO** 1AH	**then**	0101 xx00 0001 1010
17H	**LOAD** R3, [0]	R3 = 0	0001 1100 0000 0000
18H	**SUB** R3, [R2]	R3 = -Memory[addr]	1100 1110 xxxx xxxx
19H	**STORE** R3, [R2]	Memory[addr]=R3	1010 1110 xxxx xxxx
		end if	
1AH	**ADD** R2, [1]	addr = addr+1	0011 1000 0000 0001
1BH	**SUB** R1, [1]	n = n-1	0100 0100 0000 0001
1CH	**GOTO** 12H	**end loop**	0101 xx00 0001 0010
1DH	**STOP**		0000 0000 0000 0000

x's are <u>don't care</u> bits i.e. we don't care whether they are 0 or 1.

2. There are many possible designs. Here's one:

Instruction Format for

Register[N] = Memory [Address]
Memory [Address] = Register[N]

OP	Register[N]	Address
6-bits	6-bits	20-bits

Instruction Format for

Register[N] = Register[M]

OP	Register[N]	Register[M]	Unused
6-bits	6-bits	6-bits	14-bits

Instruction Format for

Register[N] = Register[M] + Register[P]
Register[N] = Register[M] - Register[P]

OP	Register[N]	Register[M]	Register[P]	Unused
6-bits	6-bits	6-bits	6-bits	8-bits

(i) For 6-bit opcode we have 64 instructions (i.e. 2^6)

(ii) 6-bits for the register number gives us 64 registers

(ii) 20-bit address gives us 1M words of memory (word-addressable)
or if memory word size = 32 bits and memory is byte addressable then
20-bit address = 1M bytes = 256Kwords of memory.

Note: This solution describes an architecture whose instructions are of the same size. Some real architectures support variable size instructions. Why?