1) A- 1 and 2 are hex codes in the registers representing the assembly code

B- compiler

C- assembler

3) issues with endianness and libraries.

4) in the final 22 bits

6) static linking is done at compile time but dynamic linking is done at run time.

8) A- memory allocation is done automatically via garbage collection

B- memory allocation has to be done manually

C- memory is not coalesced and pulls separated memory blocks it made to handle memory requests

9) value of ebp is pushed onto the stack.

Then esp moved to ebp.

Then ebp moved back to esp.

Then value popped from stack to ebp.

11) A- 8D

B- DD

C- 00

D- G4

12) 0xFF

14) 0x100

15) 0x81

1 1 1

17) A 7 9 3 5 D

+ 5 C 8 D 3

A D 5 C 3 0

18) 01000011010001011011000000000000

19) A-222

B-111

C- 333

D- 6

E- 555

F- 777

G- 4

H- 12

I- 4

20) First expression result: 12 more than the address of eax

Second result: the address that is 4 times the address of ecs plus the address of eax

Third result: the address that is 5 plus the address of eax plus 8 times the address of eax

Fourth result: the address that is 10 plus 8 times the address of ecx

23) same code as the lab we had that returns the specified byte of an integer. For me that involved shifting the integer all the way to the left till the specified byte was reached, then right till the specified byte was isolated, thus leaving us with the byte and a bunch of 0s.

24) unsigned int day(unsigned int d){

d<<20;

d>>20

Return d;

}

Unsigned int month(unsigned int d){

d<<16;

d>>28;

Return d

}

Unsigned int year (unsigned int d){

d << 11;

d >> 27;

Return d;

}

26) data pop(starting\_node){

while(starting\_node.next != null){

Starting\_node = starting\_node.next;

if(starting\_node.next.next == null){

Value = starting\_node.next.data;

Starting\_node.next = null;

}

}

Return value;

}