Majibu Nshombo
CSC 231
Due Oct, 3

Assembly Language Assignment 2

Question 1. Describe the relation between a basic assembly code and memory segmentation.

- The relation between a basic assembly code and memory segmentation is that, in assembly coding the intermediate register values are stored in the memory space and o also the memory is divided into three independent segments which are referenced by the pointers. I've learned that each of these segments stores different types of data values like instruction code, data elements and program stack. The memory in assembly code is categorized into three segments which are:
  - The Data segment: It stores the data element of the code, it remains static throughout the code, the extension used for data segment is. bss
  - The Code segment: It is the segment of memory which stores the instruction code, it has fixed area in memory, the extension used for code segment is .text
  - And the Stack: It stores the information related to passed values to the function or the procedure within the program.

Question 2. If we compile the following piece of C++ code, explain how the registers will act?

String str = "Welcome"
Std: cout<<str;

- If we compile the following piece of C++ code, I believe the std register will request to the compiler that the specified variable string str is to be stored in a register of the processor instead of memory as a way to gain speed, mostly because it will be heavily used.

Question 3. When an instruction requires two operands, what do the first operand and the second operand represent? What does the following addressing mode represent?

- When the instruction requires two operands, then the first operand (left hand) represents that source operand and the second operand (right hand) represents the destination operand. The following addressing mode represents two address instructions.

Question 4. **a) What is the syntax of the MOV instruction?**

- The MOV instruction is basically used for moving data from one storage space to another storage space. And also, the MOV instruction takes two operands. Copies the second operand (source operand) to the first operand (destination operand). According to what I've learned the source operand can be an immediate value, general-purpose register, segment register, or memory location.
  The syntax of the MOV instruction is MOV destination, source.

b) The five forms of MOV instruction:

- MOV register, register
- MOV register, immediate
- MOV memory, immediate
- MOV register, memory
- MOV memory, register

Question 5. **Find the name of the variable and variable-length from the following code** **https://rextester.com/l/nasm_online_compiler**

- The variable name is "hello", and the name of its length is "helloLen".

Question 6. **Why the two codes differ from each other** **code 1** **code2**

In the above programs that are code 1 and code 2 the only difference is that:

- In the first code is one which outputs "**Hello world!**". The first code has "**helloLen: equ $-hello**", which implies the Length of the 'Hello world!' string. The first code also has "**mov edx,helloLen**", which is a constant that's been used to get its actual value, which would then be "**Hello world**".

- While the second code outputs "**Hell**". The second code also has "**helloLen: equ $-hello**", which implies the Length of the 'Hello world!' string. But unlike the first code, the second code also has "**mov edx,4**", which is being used to get the first 4 strings from the hello variable, which would be "**Hell**".

Question 7. What are the changes you have to make in the code Code_Multiple-Initialization to get the following output. (Make changes in code or re-write. Then press the button RUN(F8) . After running save it and name it. You will be given a link, which you will copy and paste it in your assignment.)
#^##^##^##^#

- Instead of running the code in line 7, we will have to change dx 9 to dx 12.
- In line 17 we chanded 9 db with 12db, and '*' with '#^#'.

https://rextester.com/CBCV83871

Question 8. Describe each line of the following code
https://rextester.com/l/nasm_online_compiler.

Mov eax,4:

- Before invoking the system call we need to store the system call number in eax register. 4 represents write system call

mov ebx,1:

- ebx is used to choose how to display the output. 1= console output.

mov ecx, hello:

- Store the starting address of the string in ecx

mov edx, hellolen:

- store the number of bytes to write one for each letter of content strings

Int 80h:

- call the kernel (execute the system call)

mov eax, 1:

- we are now invoking exit system call, so store 1 in eax. When eax = 1, we can have ebx = 0 or 1.

mov ebx, 0:

- 0 = normal exit, 1 = error

Int 20h:

- Call the kernel (execute the system call)