Assignment 5

"I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work."

Name	Ravi Trivedi
Student Number	105197609
UWinID	trive11w@uwindsor.ca

Section 1

Question 1:

1. Which instruction pushes all of the 32-bit general-purpose registers on the stack? (1 point)
    - PUSHAD
2. Which instruction pushes the 32-bit EFLAGS register on the stack? (1 point)
    - PUSHFD
3. Which instruction pops the stack into the EFLAGS register? (1 point)
    - POPFD
4. What will be the final value in EAX after these instructions execute? (2 points)
   push 5
   push 6
   pop eax
   pop eax
    - The final answer in the EAX will be 5, because 6 will be popped out in line 3.
5. The stack size can be changed at run time. (T/F) (2 points)
    - False. After running the program, the stack size is already determined and thus cannot be changed.
6. Suppose there were no PUSH instruction. Write a sequence of two other instructions that would accomplish the same as push eax. (3 points)
    - SUB ESP, 4
    - MOV [ESP] , EAX
    - These steps will work exactly as PUSH instruction.

Question 2 :

1. An assembler (called NASM) permits the PUSH instruction to list multiple specific registers. Why might this approach be better than the PUSHAD instruction in MASM? Here is a NASM example: (4 points)
   PUSH EAX EBX ECX
    - This approach is better because PUSHAD pushes all the registers, while PUSH will only push one register. That waste stack memory as PUSHAD will push all 8 32-bit registers, means stack will store less. Meanwhile, using PUSH, we have to do line by line push and is repetitive.

2. Discuss the Runtime Stack using an example and briefly explain the operations which can be performed on this stack. (4 points)
    - Stack works on the LIFO(last in first out) structure, meaning whatever comes last is the first one to comes out. Stack has two registers. One for stack pointer which is ESP and other for stack segment which is SS. ESP is pointed to the top element of the stack.
    - There are many operations we can use in stack, like PUSH for adding a new element to the stack, POP is for popping the top element off the stack, RET pops the top element and copies it to the EIP.
    - Example :
    - push 1 ; pushes 1 to stack
    - push 2 ; pushes 2 to stack
    - pop eax; pops the last element which is 2 to eax
3. Briefly explain what a Stack Overflow is in a system stack. (4 points)
    - Stack overflow happens when the too much is pushed by computer program to stack. Using too much PUSH instruction causes ESP to drop below the minimum address line, and thus an error occurs.
4. Explain using an example how the stack push operation works. (4 points)
    - The push operation adds the new element to the stack and decreasing stack pointer.
    - Example of the push is :
    - push 5 ; pushes 5 to stack
5. Explain using an example how the CALL and RET instructions work. (4 points)
    - The CALL instruction copies the value from EIP and place to it stack.
    - The RET instruction copies the value that has been popped by stack and copies it to EIP.
    - Example :

```
call_ret  PROC
        ; some code
        ret ; pops the top element off the stack and copies it to EIP
call_ret ENDP

main PROC
        call call_ret ; copies the value from EIP to stack
        exit
main ENDP
```