WORKING WITH ASSEMBLY LANGUAGE AND UNDERSTANDING PROCESS SEGMENTS

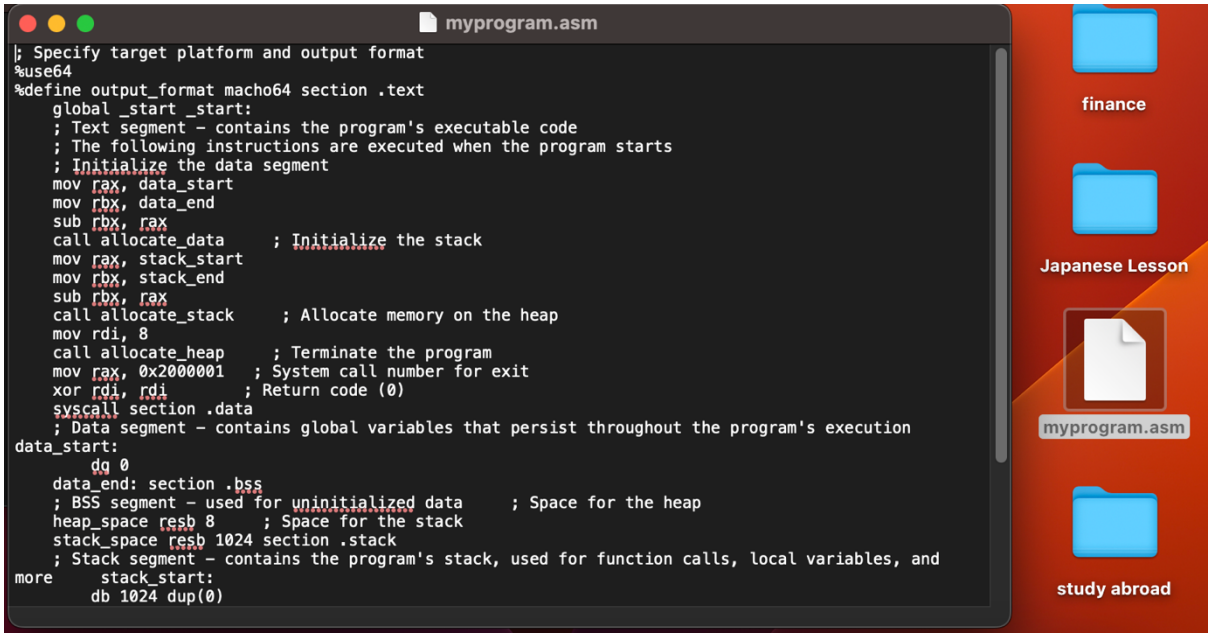
1. INTRODUCTION

In this report, we execute and observe code written in assembly language and C#. Also we examine what these codes are trying to perform.

2. RESULT OF EXECUTING ASSEMBLY CODE

I tried to compile the assembly code, but an error occurred, and I could not compile, link, and run it.

Here is the assembly code.




```
; Specify target platform and output format
%use64
%define output_format macho64 section .text
    global _start _start:
    ; Text segment - contains the program's executable code
    ; The following instructions are executed when the program starts
    ; Initialize the data segment
    mov rax, data_start
    mov rbx, data_end
    sub rbx, rax
    call allocate_data     ; Initialize the stack
    mov rax, stack_start
    mov rbx, stack_end
    sub rbx, rax
    call allocate_stack     ; Allocate memory on the heap
    mov rdi, 8
    call allocate_heap     ; Terminate the program
    mov rax, 0x2000001   ; System call number for exit
    xor rdi, rdi        ; Return code (0)
    syscall section .data
    ; Data segment - contains global variables that persist throughout the program's execution
data_start:
        dq 0
    data_end: section .bss
    ; BSS segment - used for uninitialized data     ; Space for the heap
    heap_space resb 8     ; Space for the stack
    stack_space resb 1024 section .stack
    ; Stack segment - contains the program's stack, used for function calls, local variables, and
more     stack_start:
        db 1024 dup(0)
```

Here is the error message.

```
Desktop — -zsh — 100×27
Last login: Sun Feb 19 13:55:51 on ttys000
misatoseki@MisatonoMacBook-Air ~ % cd Desktop
misatoseki@MisatonoMacBook-Air Desktop % nasm -f macho64 myprogram.asm -o myprogram.o
myprogram.asm:2: error: label or instruction expected at start of line
myprogram.asm:5: warning: label alone on a line without a colon might be in error [-w+label-orphan]
myprogram.asm:6: warning: label alone on a line without a colon might be in error [-w+label-orphan]
myprogram.asm:7: warning: label alone on a line without a colon might be in error [-w+label-orphan]
myprogram.asm:9: error: label `   ' inconsistently redefined
myprogram.asm:4: info: label `   ' originally defined here
myprogram.asm:10: error: label `   ' inconsistently redefined
myprogram.asm:9: info: label `   ' originally defined here
myprogram.asm:11: error: label `   ' inconsistently redefined
myprogram.asm:10: info: label `   ' originally defined here
myprogram.asm:12: error: label `   ' inconsistently redefined
myprogram.asm:11: info: label `   ' originally defined here
myprogram.asm:13: error: label `   ' inconsistently redefined
myprogram.asm:12: info: label `   ' originally defined here
myprogram.asm:14: error: label `   ' inconsistently redefined
myprogram.asm:13: info: label `   ' originally defined here
myprogram.asm:15: error: label `   ' inconsistently redefined
myprogram.asm:14: info: label `   ' originally defined here
myprogram.asm:16: error: label `   ' inconsistently redefined
myprogram.asm:15: info: label `   ' originally defined here
myprogram.asm:17: error: label `   ' inconsistently redefined
myprogram.asm:16: info: label `   ' originally defined here
myprogram.asm:18: error: label `   ' inconsistently redefined
myprogram.asm:17: info: label `   ' originally defined here
```

3. RESULT OF EXECUTING C# CODE

The C# code worked fine and was completed.

```
Program.cs ×
Program.cs
1   using System;
2
3   namespace AssemblyExample
4   {
5       class Program
6       {
7           static void Main(string[] args)
8           {
9               // Allocate memory on the heap
10              byte[] heapMemory = new byte[8];
11
12              // Terminate the program
13              Environment.Exit(0);
14          }
15      }
16  }

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

misatoseki@MisatonoMacBook-Air myprogram % dotnet run
misatoseki@MisatonoMacBook-Air myprogram %
```

4. WHAT THESE CODES ARE TRYING TO PERFORM

Both assembly code and C# cod are a program that performs program segments. Process segmenting is one method of partitioning the memory used during program execution, a feature that enables efficient program execution by dividing the memory space of a process into multiple segments.

Specifically, both code allocate 8 bytes on the heap memory and then call the Environment.Exit(0) method to terminate the program.

5. WHAT I LEARNED

I found that Both assembly code and C# code say same thing. However, assembly language, which is low-level, or near-machine language, has more code than C#.

In this report, my machine couldn't compile the assembly code. Does it matter that I used M1 Mac to execute the code? We were not able to find out anymorein the class time.