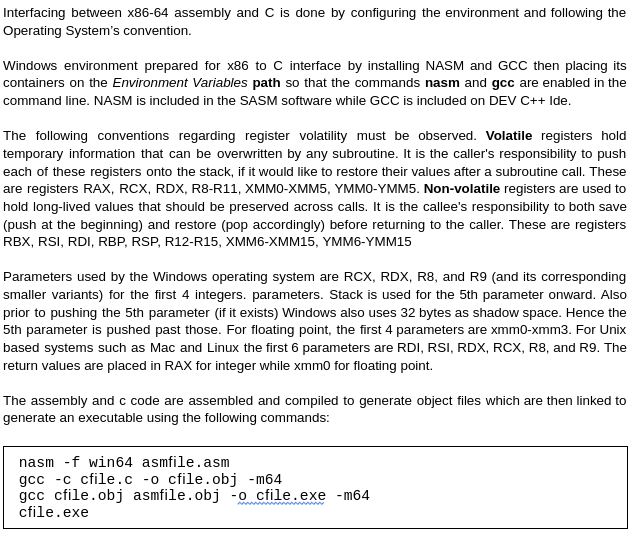
|  |  |
| --- | --- |
|  | De La Salle University Computer Technology Department  Laboratory for Computer Organization and Architecture |

**Module 9: C to x86 Interfacing**

|  |  |  |  |
| --- | --- | --- | --- |
| Names | Andrei D. Palonpon | Date Performed | 11 / 21 / 2024 |
| Daniel Gavrie Y. Clemente |  |

**I. Introduction**



**II. Programming Exercises:**

Place your code on the box provided. Provide a screenshot that your code works.

1. Write an x86-64 bit assembly program that calls the C’s printf function to display the message and corresponding data declared in the data segment as.   
     
   msg dd “seven numbers %d, %d, %d, %d, %d, %d, %d”,10,13,0   
   a dd 1   
   b dd 2   
   c dd 3   
   d dd 4   
   e dd 5   
   f dd 6   
   g dd 7   
     
    Use SASM’s save as .exe to generate the executable file.   
     
   X86 assembly code

|  |
| --- |
| section .data  msg dd "seven numbers %d, %d, %d, %d, %d, %d, %d", 10, 13, 0  a dd 1  b dd 2  c dd 3  d dd 4  e dd 5  f dd 6  g dd 7  section .text  bits 64  default rel  global main  extern printf  main:  mov rbp, rsp; for correct debugging  sub rsp, 8\*8    ;Regardless of data types and their sizes, it has to be 8-byte aligned.  mov rax, 0  mov rcx, 0  mov rdx, 0  mov r8, 0  mov r9, 0    mov rax, [g] ;param 7: g  mov [rsp+56], rax    mov ax, [f] ;param 6: f  mov [rsp+48], rax    mov ax, [e] ;param 5: e  mov [rsp+40], rax    mov ax, [d] ;param 4: d  mov [rsp+32], rax    mov r9, [c] ;param 3: c  mov r8, [b] ;param 2: b  mov rdx, [a] ;param 1: a  lea rcx, [msg]  call printf  add rsp, 8\*8    xor rax, rax  ret |

Screenshot

|  |
| --- |
|  |

1. Write a C and an x86 assembly program that asks for two number inputs and subtracts the first input by the second input. The C code is responsible for collecting and displaying the inputs and outputs. While the x86 assembly code is responsible for the subtraction. The C function prototype should be  
     
   extern int asmsub(int a, int b);   
     
   X86 assembly code

|  |
| --- |
| section .text  bits 64  default rel  global asmsub  asmsub:  ;write your code here  ;a@rcx, b@rdx  mov rax, rcx  sub rax, rdx    ret |

C code

|  |
| --- |
| #include <stdio.h>  extern int asmsub(int a, int b);  int main()  {  int a, b;  scanf("%d %d", &a, &b);  printf("%d", asmsub(a, b));  return 0;  } |

Assembling, Compiling, and Linking Commands

|  |
| --- |
|  |

Screenshot

|  |
| --- |
|  |

1. Write a C and an x86 assembly program that subtracts the values from two input vectors   
     
   int vec1[] = {20, 30, 40 50};   
   int vec2[] = {5, 10, 15, 20};   
     
   place the result into vec3 then display the contents of vec3 The C code is responsible for allocating spaces, declaration, and initialization of the vectors. While the x86 assembly program is responsible for subtracting the vectors. The C function prototype should be   
     
   extern int asmvecsub(int n, int\* vec1, int\* vec2, int\* vec3);

X86 assembly code

|  |
| --- |
| section .text  bits 64  default rel  global asmvecsub  asmvecsub:  ;n@rcx, vec1@rdx, vec2@r8, vec3@r9  L1:  mov rax, [rdx]  mov rbx, [r8]  sub rax, rbx  mov [r9], rax  add rdx, 4  add r8, 4  add r9, 4  loop L1    ret |

C code

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  extern int asmvecsub(int n, int \*vec1, int \*vec2, int \*vec3);  int main()  {  int vec1[] = {20, 30, 40, 50},  vec2[] = {5, 10, 15, 20},  n = 4,  \*vec3 = (int \*)malloc(n \* sizeof(int));  for (int i = 0; i < n; i++)  printf(" %d", vec3[i]);  printf("\n");  return 0;  } |

Assembling, Compiling, and Linking Commands

|  |
| --- |
|  |

Screenshot

|  |
| --- |
|  |