

hw6: Assembly Language 2

Due Apr 17 at 11:59pm

Points 8

Questions 8

Available Apr 5 at 12am - Apr 18 at 11:59pm

Time Limit 80 Minutes

Allowed Attempts 3

Instructions

Covers Stack Frames and Function Calls

Suggestions:

- Use your knowledge of what the compiler adds to functions to determine what parts of the code are most important. For example, questions that require code tracing don't necessarily require every assembly instruction to be traced.
- Some parts of assembly code are underlined to highlight details to be noticed.
- Recall in C, the operator >> is bit shift right and << is bit shift left.
- You may use the x86-cheat-sheet.pdf found in the Files section on course site for a reference of x86 assembly instructions.
- **Note:** The cheat sheet will NOT be available for the final exam but familiarizing yourself with the assembly instructions and their format will help you interpret any reference material that may be provided for specific questions.

This quiz was locked Apr 18 at 11:59pm.

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	23 minutes	8 out of 8
LATEST	Attempt 2	23 minutes	8 out of 8
	Attempt 1	79 minutes	3.5 out of 8

Score for this attempt: 8 out of 8

Submitted Apr 17 at 8:30pm

This attempt took 23 minutes.

Question 1

1 / 1 pts

```

<doubleIt>:
    pushl %ebp
    movl %esp, %ebp
    pushl %ebx
    pushl %esi
    movl X(%esp), %ecx
    addl %ecx, %ecx
    movl %ecx, %eax
    popl %ecx
    popl %ebx
    leave
    ret

```

```

int doubleIt(int x){
    return x + x;
}

```

The value of **X** that would result in the correct execution of `doubleIt` is:

Correct!

16

Correct Answers

16 (with margin: 0)

Question 2

1 / 1 pts

Consider the following assembly code:

```

movl 0x8(%ebp),%eax
movl (%eax),%edx
movl 0xc(%ebp),%eax
movl (%eax),%eax
addl %edx,%eax
movl 0xc(%ebp),%edx
movl %eax,(%edx)

```

If `0x8(%ebp)` and `0xc(%ebp)` refer to the function arguments `a` and `b` respectively, which one of the following choices represent the C equivalent of the assembly code above?

☐

```
void func(int* a, int* b) {  
    int tmp = *b;  
    *b = *a;  
    *a = tmp;  
    return;  
}
```

☐

```
void func(int* a, int* b) {  
    a = b + a;  
    return;  
}
```

☐

```
void func(int* a, int* b) {  
    b = a + b;  
    return;  
}
```

☐

```
void func(int* a, int* b) {  
    *a = *a + *b;  
    return;  
}
```

☒

```
void func(int* a, int* b) {  
    *b = *a + *b;  
    return;  
}
```

Correct!

Question 3

1 / 1 pts

If P calls $G(x, y)$, then which of the following statements below are correct ?

1. Arguments passed to G are stored in G's stack frame.
2. Return address to P is stored in G's stack frame.
3. G can safely overwrite %eax, %edx or %ecx without saving their data first.
4. G cannot safely overwrite %ebx, %esi or %edi without saving their data first.

Correct!

- ☐ 1 and 3
- ☐ 1, 2, and 3
- ☒ 3 and 4
- ☐ 2 and 3
- ☐ 1, 2, 3, and 4
- ☐ 2 and 4

Question 4

1 / 1 pts

```
<func>:
    pushl %ebp
    movl %esp, %ebp
    subl $16, %esp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $0, %eax
    cmpl %ecx, %edx
    jge L1
    movl $1, %eax
L1:
    leave
    ret
```

Which one of the comparisons below does the call `func(x, y)` compute?

- ☐ `y <= x`
- ☒ `y < x`
- ☐ `x < y`
- ☐ `x <= y`
- ☐ `y == x`

Correct!

Question 5

1 / 1 pts

Consider the following function where **X**, **Y**, **Z**, and **W** represent incomplete code.

```
int math(int a, int b, int c) {  
    int res1 = X;  
    int res2 = Y;  
    int res3 = Z;  
    int res4 = W;  
    return res4;  
}
```

The function above is implemented in assembly code below where the function's parameters *a*, *b* and *c* are at the effective addresses `0x8(%ebp)`, `0xC(%ebp)` and `0x10(%ebp)`, respectively.

```
movl 0xC(%ebp),%eax  
notl %eax  
notl %eax  
movl %eax,-0x10(%ebp)  
movl 0x10(%ebp),%eax  
movl 0xC(%ebp),%edx  
addl %edx,%eax  
movl %eax,-0xC(%ebp)  
movl 0x10(%ebp),%eax  
imull $11,%eax  
movl %eax,-0x8(%ebp)  
movl -0xC(%ebp),%eax  
sall $0x5,%eax  
movl %eax,-0x4(%ebp)
```

Choose the correct options for **X**, **Y**, **Z** and **W** to complete the function so that it corresponds with its assembly code.

- **X** is
- **Y** is
- **Z** is
- **W** is

Answer 1:

Correct!

b

Correct!

Answer 2:

b + c

Correct!

Answer 3:

c * 11

Correct!

Answer 4:

res2 << 5

Question 6

1 / 1 pts

Consider the following funtion:

```
int recursive(int n) {...}
```

The assembly code equivalent of the above function is:

```
recursive:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    subl $0x14,%esp
    cmpl $0x1,0x8(%ebp)
    je .L1
    cmpl $0x2,0x8(%ebp)
    jne .L2
.L1:
    movl 0x8(%ebp),%eax
    jmp .L3
.L2:
    movl 0x8(%ebp),%eax
    subl $0x1,%eax
    movl %eax,(%esp)
    call recursive
    movl %eax,%ebx
    movl 0x8(%ebp),%eax
    subl $0x2,%eax
    movl %eax,(%esp)
    call recursive
    imul %ebx,%eax
.L3:
    addl $0x14,%esp
    popl %ebx
    popl %ebp
    ret
```

What would be the values returned for the code below?

```
int ret_val_1 = recursive(1);
int ret_val_2 = recursive(2);
```

☐ ret_val_1 is 0 and ret_val_2 is 1

☐ ret_val_1 is 1 and ret_val_2 is 1

☒ ret_val_1 is 1 and ret_val_2 is 2

☐ ret_val_1 is 1 and ret_val_2 is 0

☐ ret_val_1 is 2 and ret_val_2 is 1

Correct!

Question 7

1 / 1 pts

Consider the following funtion:

```
int recursive(int n) {...}
```

The assembly code equivalent of the above function is:

```
recursive:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    subl $0x14,%esp
    cmpl $0x1,0x8(%ebp)
    je .L1
    cmpl $0x2,0x8(%ebp)
    jne .L2
.L1:
    movl 0x8(%ebp),%eax
    subl $1,%eax
    jmp .L3
.L2:
    movl 0x8(%ebp),%eax
    subl $0x1,%eax
    movl %eax,(%esp)
    call recursive
    movl %eax,%ebx
    movl 0x8(%ebp),%eax
    subl $0x2,%eax
    movl %eax,(%esp)
    call recursive
    addl %ebx,%eax
.L3:
    addl $0x14,%esp
    popl %ebx
```

```
popl %ebp
ret
```

What would be the value returned by the code below?

```
int return_val = recursive(5);
```

(Hint: Consider determining and then tracing the equivalent code in C.)

☐ 4

☐ 8

☐ 6

☒ 3

☐ 10

Correct!

Question 8

1 / 1 pts

```
<func>:
    pushl %ebp
    movl %esp, %ebp
    subl $16, %esp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $0, %eax
    cmpl %ecx, %edx
    jle L1
    movl $1, %eax
L1:
    leave
    ret
```

Which one of the following assembly code fragments is equivalent to the above assembly code fragment?

Correct!

☐

```
<func>:
    pushl %ebp
    movl %esp, %ebp
    subl $16, %esp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $0, %eax
    subl %ecx, %edx
    cmpl %edx, $0
    jl L1
    movl $1, %eax
L1:
    leave
    ret
```

☒

```
<func>:
    pushl %ebp
    movl %esp, %ebp
    subl $16, %esp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $0, %eax
    subl %ecx, %edx
    cmpl %edx, $0
    jge L1
    movl $1, %eax
L1:
    leave
    ret
```

☐

```
<func>:
    pushl %ebp
    movl %esp, %ebp
    subl $16, %esp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $1, %eax
    subl %ecx, %edx
    cmpl %edx, $0
    jge L1
    movl $0, %eax
L1:
    leave
    ret
```

☐ `<func>:`
 `pushl %ebp`
 `movl %esp, %ebp`
 `subl $16, %esp`
 `movl 8(%ebp), %ecx`
 `movl 12(%ebp), %edx`
 `movl $0, %eax`
 `subl %edx, %ecx`
 `cmpl %ecx, $0`
 `jge L1`
 `movl $1, %eax`

L1:
 `leave`
 `ret`

☐ `<func>:`
 `pushl %ebp`
 `movl %esp, %ebp`
 `subl $16, %esp`
 `movl 8(%ebp), %ecx`
 `movl 12(%ebp), %edx`
 `movl $0, %eax`
 `subl %edx, %ecx`
 `cmpl %ecx, $0`
 `j1 L1`
 `movl $1, %eax`

L1:
 `leave`
 `ret`

Quiz Score: **8** out of 8