Assignment#7 Part 2: x86 assembly

CS232 Spring 2021

Due: Monday, April 26 at 11:59:59am

Elijah Goldstein

Notes

- Please choose “File”-> “Make a copy” to create a copy of this google document under your google account, and fill in your answers in your own copy, because you do not have permission to edit this document in place.
- All the answer text areas are already set in blue. Please try to keep the blue setting for your answer text. Thanks for your collaboration in helping me with streaming grading.
- Once you are ready to submit your homework, choose “File” -> “Download” -> “PDF Document” to save your homework locally as a pdf file.
- In exams you have no access to compiler explorer, so you are recommended to solve the problems without it. You can use the compiler to verify your solution afterwards but please try not to rely on it while doing your homework.

---

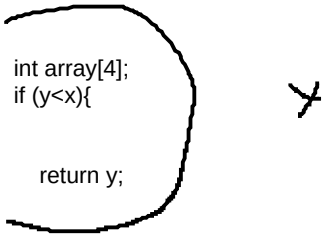
1. **[8 points]** Write C code for `func()` based on the following assembly code that was generated.

```
func:
      pushl   %ebp
      movl  %esp, %ebp
      subl  $16, %esp
      movl  8(%ebp), %eax
      cmpl  12(%ebp), %eax
      jle .L2
      movl  8(%ebp), %eax
      movl  %eax, -4(%ebp)
      jmp .L3
.L2:
      movl  12(%ebp), %eax
      movl  %eax, -4(%ebp)
.L3:
      movl  -4(%ebp), %eax
      leave
      ret
main:
      pushl   %ebp
      movl  %esp, %ebp
      subl  $16, %esp
```

```
        pushl   $20
        pushl   $10
        call  func
        addl  $8, %esp
        movl  %eax, -4(%ebp)
        movl  $0, %eax
        leave
        ret
int func(int x, int y) {
//write your code here

}

int main(){
        int rc = func(10, 20);
        return 0;
}
```



2. **[8 points]** A function with prototype

```
int decode2(int x, int y, int z);
```

is compiled into 32bit x86 assembly code. The body of the code is as follows:

```
NOTE: x at %ebp+8, y at %ebp+12, z at %ebp+16

movl 12(%ebp), %edx
subl 16(%ebp), %edx
movl %edx, %eax
sall $31, %eax
sarl $31, %eax
imull 8(%ebp), %edx
xorl %edx, %eax
```

Parameters x, y, and z are stored at memory locations with offsets 8, 12, and 16 relative to the address in register %ebp. The code stores the return value in register %eax. The shl or sal instruction is used to shift the bits of the operand destination to the left, by the number of bits specified in the count operand

Write C code for `decode2` that will have an effect equivalent to our assembly Code.

```
int decode2(int x, int y, int z) {
//write your code here:
}
```

3. Consider the following assembly code for a C **`for`** loop: **[10 points]**

```
loop:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%ecx
        movl 12(%ebp),%edx
        xorl %eax,%eax
        cmpl %edx,%ecx
        jle .L4
.L6:
        decl %ecx
        incl %edx
        incl %eax
        cmpl %edx,%ecx
        jg .L6
.L4:
        incl %eax
        movl %ebp,%esp
        popl %ebp
        ret
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: you may only use the symbolic variables **`x`**, **`y`**, and **`result`** in your expressions below --- **do NOT use register names**.

```
int loop(int x, int y)
{
   int result;

   for (  x < y          ;  ___________; result++ ) {

      _________________;

      _________________;
   }

   _________________;

   return result;
}
```

```
int loop(int x, int y)
{
   int result;
   for (x<y;result=0; result++)
   {
      if (x<result){
         x++;
         y--;
      }
   }
result++;

 return result;

}
```

4. **[8 points]** Match the following C functions (C1, C2, C3 and C4) with their corresponding assembly functions (A1, A2, A3, and A4). Write your answers in the spaces provided at the end of the questions.

C program	Assembly Program

C1
 	
`int func(int x, int y)`
`{`
`  int result = x && y;`
`  return result;`
`}`
	A1
	
	`func:`
	`  pushl %ebp`
	`  movl  %esp, %ebp`
	`  subl  $16, %esp`
	`  movl  8(%ebp), %eax`
	`  orl  12(%ebp), %eax`
	`  movl  %eax, -4(%ebp)`
	`  movl  -4(%ebp), %eax`
	`  leave`
	`  ret`

C2
 	
`int func(int x, int y)`
`{`
`  int result = x || y;`
`  return result;`
`}`
	A2
	
	`func:`
	`  pushl %ebp`
	`  movl  %esp, %ebp`
	`  subl  $16, %esp`
	`  cmpl  $0, 8(%ebp)`
	`  je  .L2`
	`  cmpl  $0, 12(%ebp)`
	`  je  .L2`
	`  movl  $1, %eax`
	`  jmp .L3`
	`.L2:`
	`  movl  $0, %eax`
	`.L3:`
	`  movl  %eax, -4(%ebp)`
	`  movl  -4(%ebp), %eax`
	`  leave`
	`  ret`

C3
 	
`int func(int x, int y)`
`{`
`  int result = x & y;`
`  return result;`
	A3
	
	`func:`
	`  pushl %ebp`
	`  movl  %esp, %ebp`
	`  subl  $16, %esp`

`}`	`movl  8(%ebp), %eax`
	`andl  12(%ebp), %eax`
	`movl  %eax, -4(%ebp)`
	`movl  -4(%ebp), %eax`
	`leave`
	`ret`

C4
 	
`int func(int x, int y)`
`{`
`    int result = x | y;`
`    return result;`
`}`
	A4
	
	`func:`
	`  pushl %ebp`
	`  movl  %esp, %ebp`
	`  subl  $16, %esp`
	`  cmpl  $0, 8(%ebp)`
	`  jne .L2`
	`  cmpl  $0, 12(%ebp)`
	`  je  .L3`
	`.L2:`
	`  movl  $1, %eax`
	`  jmp .L4`
	`.L3:`
	`  movl  $0, %eax`
	`.L4:`
	`  movl  %eax, -4(%ebp)`
	`  movl  -4(%ebp), %eax`
	`  leave`
	`  ret`

Write your answers below: (If C1 matches with A4, write A4 in the space next to C1)

C1 - A3

C2 - A1

C3 - A2

C4 - A4

5. **[16 points]** Consider the following recursive factorial function in C and Assembly language.

```
int rfact(int n)
{
    int result;
    if (n <= 1)
        result = 1;
    else
        result = n * rfact(n-1);
```

```
    return result;
}
```

```
Line#  Assembly Code
   1.   rfact:
   2.       pushl %ebp
   3.       movl %esp, %ebp
   4.       pushl %ebx
   5.       subl $4, %esp
   6.       movl 8(%ebp), %ebx
   7.       movl $1, %eax
   8.       cmpl $1, %ebx
   9.       jle .L53
   10.      leal -1(%ebx), %eax
   11.      movl %eax, (%esp)
   12.      call rfact
   13.      imull %ebx, %eax
   14.   .L53:
   15.      addl $4, %esp
   16.      popl %ebx
   17.      popl %ebp
   18.      ret
```

Questions:

1. Why do we push the %ebx register's value on the stack frame of rfact?
   (Refer: Line# 4 in assembly code - `pushl %ebx`)

   Your answer:
   pushl pushes source operand onto stack
   its pushing %ebx register value for variable n in order to do the calculation in the else statement to solve for the value of result.

2. What is the purpose of the following 2 statements?
   a. `subl $4, %esp (Line number 5)`

      Your answer:
      this is demonstration the creation of a data type of 4 bytes being allocated to type int for the creation of int result;
      subl subtracts second operand from first and stores it in first operand known as the destination ($4)

   b. `addl $4, %esp (Line number 15)`

      Your answer:

3. For every invocation of the function rfact( ) which register is used to store the value of its input

argument?

Your answer: %ebx

4. What is the purpose of the following line of assembly code (Line number 10)?

```
leal -1(%ebx), %eax
```

Your answer:
leal will get memory address of %eax register and put it in the %ebx register

5. Why are the following 2 lines (Line numbers 2 - 3) needed in `rfact()` function?

```
pushl %ebp
movl %esp, %ebp
```

Your answer:
it pushes the rfact int function onto stack and its parameters
(pushl decrements stack pointer and stores source on top of stack while movl moves %ebp to the stack pointer %esp

---

Credits: Thanks Prof. Remzi H. Arpaci-Dusseau for some task ideas and materials. The tasks are edited and modified to suit the needs of CS232 Spring 2020 at Pace by Dr. Jun Yuan.