


Assembly and Machine Language - Spring 1397 (2018) Final Exam	Instructor: B. Nasihatkon	
Name:	ID:	Khordad 1397 - June 2018

- All programs are written in the 32-bit mode on an x86 platform.

Question 1 The C function on the left receives an array of integers plus its length as arguments and returns the sum of squared elements of the array. Complete the assembly program on the right to call the `sumsq` function on the array "array" defined in the data segment, and then print the computed sum of squared elements using the `printf` function from the C standard library. You are not allowed to use the `print_int` function. (20 points)

```

#include <stdio.h>

int sumsq(int a[], int n) {
    int s = 0;

    for (int i = 0; i < n; i++)
        s += a[i] * a[i];

    return s;
}

```

label	command	arguments
segment .data		
array:	<code>dd</code>	<code>1, 2, -2, 4, -3, 2</code>
segment .text		
extern global		
main:		
	<code>mov</code>	<code>eax, 1</code>
	<code>int</code>	<code>0x80</code>

Question 2 Here, we do the opposite of what was done in Question 1. Now, the sumsqr function is written in assembly and is called from C. On the left you can see how the sumsqr function is called. Complete the assembly code on the right to write the body of the sumsqr function. Use appropriate directives (global, extern, etc.) if needed. **Observe all C calling conventions.** (20 points)

<pre style="margin: 0;">#include <stdio.h> extern int sumsqr(int a[],int n); int main() { int array[6] = {1,2,-2,4,-3,2}; int s = sumsqr(array,6); printf("%d\n", s); return 0; }</pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">label</th> <th style="width: 30%;">command</th> <th style="width: 50%;">arguments</th> </tr> </thead> <tbody> <tr> <td colspan="3" style="padding: 5px;"><code>segment .text</code></td> </tr> <tr> <td style="padding: 5px;"><code>sumsqr:</code></td> <td></td> <td></td> </tr> <tr> <td></td> <td style="padding: 5px;"><code>push</code></td> <td style="padding: 5px;"><code>ebp</code></td> </tr> <tr> <td></td> <td style="padding: 5px;"><code>mov</code></td> <td style="padding: 5px;"><code>ebp, esp</code></td> </tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr> <td style="padding: 5px;"><code>loop1:</code></td> <td style="padding: 5px;"><code>lodsd</code></td> <td></td> </tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr> <td></td> <td style="padding: 5px;"><code>loop</code></td> <td style="padding: 5px;"><code>loop1</code></td> </tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> </tbody> </table>	label	command	arguments	<code>segment .text</code>			<code>sumsqr:</code>				<code>push</code>	<code>ebp</code>		<code>mov</code>	<code>ebp, esp</code>																						<code>loop1:</code>	<code>lodsd</code>																		<code>loop</code>	<code>loop1</code>															
label	command	arguments																																																																							
<code>segment .text</code>																																																																									
<code>sumsqr:</code>																																																																									
	<code>push</code>	<code>ebp</code>																																																																							
	<code>mov</code>	<code>ebp, esp</code>																																																																							
<code>loop1:</code>	<code>lodsd</code>																																																																								
	<code>loop</code>	<code>loop1</code>																																																																							

Question 3 (20 points)

```
segment .text

f:
    push ebp
    mov  ebp, esp

    mov ecx, [ebp+8]
    cmp ecx, 0
    jg  recur
    mov eax, 1
    jmp endl

recur:
    dec ecx
    push ecx
    call f
    add esp, 4

    shl eax, 1

endl:

    pop ebp
    ret
```

A) Analyse the assembly function `f` defined on the left. What does it compute? Ignore the empty boxes for now. (12 points)

B) Change the function `f` so it can be called from within a C program (like below). Fill in the empty boxes in the assembly program to do so. If no changes are needed, mention this explicitly in the corresponding box. (8 points)

```
#include <stdio.h>

extern int f(int n);

int main() {

    printf("%d\n", f(10));

    return 0;
}
```

Question 4 Consider a 3D vector $u = [x, y, z]$. The size of the vector is equal to

$\|u\| = \sqrt{x^2 + y^2 + z^2}$. The corresponding normalized vector is

$\bar{u} = u/\|u\| = [x/\|u\|, y/\|u\|, z/\|u\|]$. In the following assembly program, the vector u has been stored in data segment as double precision floating point variables (in addresses u , $u+8$ and $u+16$). Write an assembly program to normalize the vector u . The normalization must be done in-place, that is the normalized vector \bar{u} must also get stored in addresses u , $u+8$ and $u+16$. (25 points)

