Assembly and Machine Language - Fall 96 Final Exam	**Instructor: B. Nasihatkon**	دانشگاه صنعتی خواجه نصیرالدین طوسی K. N. TOOSI UNIVERSITY OF TECHNOLOGY ۱۳۰۷
Name:	**ID:**	**Dey 1396 - Jan. 2018**

1- What does the following piece of code do? Explain your answer. **(10 points)**

```
segment .data
array1:     dd  1,3,6,10,16,22,28
array2:     dd  0,0,0,0,0,0

segment .text
     mov esi, array1
     mov edi, array2
     mov ecx, 6
     cld

     lodsd
     mov ebx, eax
loop1:
     lodsd
     sub eax, ebx
     stosd

     add ebx, eax

     loop loop1
```

2- We want to write the above as an assembly function `arrfunc` to be **called from C**. Below is a sample C code calling `arrfunc`. Complete the assembly code in the next page to fully implement the function. Observe all required conventions. **(12 points)**

```
#include <stdio.h>
extern void arrfunc(int array1[],
              int array2[], int n);
int main() {
  int array1[7] = {1,3,6,10,16,22,28};
  int array2[6];

  arrfunc(array1, array2, 7);

  for (int i = 0; i < 6; i++)
    printf("%d, ", array2[i]);
  putchar('\n');
  return 0;
}
```

Only write inside designated boxes:

```
segment .text



arrfunc:
     push ebp
     mov  ebp, esp



     cld
     lodsd
     mov  ebx, eax
loop1:
     lodsd
     sub  eax, ebx
     stosd
     add  ebx, eax
     loop loop1
```

3- The distance between two 2D points **u = [u_1 u_2]** and **v = [v_1 v_2]** is given by **dist(u, v) = sqrt((u_1-v_1)^2 + (u_2-v_2)^2).** We represent **u** and **v** by arrays of size two with single precision floating point elements:

```
segment .data
u:     dd  1.0, 7.0
v:     dd  4.0, 3.0
```

Write a piece of assembly code to compute the distance between vectors **u** and **v**. The distance must be stored in ST0 when your program finishes. Notice that the values above (1.0, 7.0, 4.0, 3.0) are just for illustration. You just have access to labels **u** and **v**. Do not make any assumption about the values stored in data segment. (13 points)

command	arguments

command	arguments

4- What does the function **`myfunc`** do? Explain your response. The program uses **`print_char`** from the book printing the character whose ascii code is in **`al`**. **(10 points)**

```
myfunc:
     mov al, [ecx]

     cmp al, 0
     je endfunc

     push eax
     push ecx

     inc ecx
     call myfunc

     pop ecx
     pop eax

     call print_char
endfunc:
     ret
```

5- Assume that the state of coprocessor stack is as follows. Write an assembly program that computes (a-d) * (b-d) / sqrt(c-d) and stores it in ST0. You can only use coprocessor stack registers, and are not allowed to use memory. **You need to write the state of stack after each command. Notice that the maximum stack size is 8. (15 points)**

command	arguments	stack (initially: **a,b,c,d**)

ST0	a
ST1	b
ST2	c
ST3	d
:	:

6- We want to write a function **median** computing the median of an array of integers by sorting the array and then returning the middle entry. Here is how **median** is used:

```
int main() {
  int a[] = {1,18,2,3,4,5,7,7,4,19,18,12,130};
  int n = sizeof(a) / sizeof(a[0]);

  printf("median=%d\n", median(a,n) );

  return 0;
}
```

A) First, we use **bubble sort** to sort the array. A C code has been written below. Your job is to write an **equivalent** assembly code. You may use the stack or registers to implement local variables, but cannot use data segment. **(20 Points)**

```
void bubblesort(int a[],int n) {

  for (int m = n-1; m > 0; m--)
    for (int i = 0; i < m; i++)
      if (a[i] > a[i+1]) {
        int t  = a[i];
        a[i]   = a[i+1];
        a[i+1] = t;
      }

}
```

label	command	arguments
bubblesort:	push	EBP
	mov	EBP, ESP

label	command	arguments

B) Now, we use the bubble sort function to compute the median. Look at the C code below. We want the original array `a` unchanged, so we define a new array `b` and copy the contents of `a` into `b`. The size of array `b` is dynamically determined (during run time), since it is equal to `n` which is given as an argument. **You need to allocate space for `b` on the stack**. Write the equivalent assembly code to implement the C function. By doing so, you will learn how the C compiler implements arrays with dynamic size. **(20 points)**

```
int median(const int a[], int n)
{
  int b[n];

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

  bubblesort(b,n);

  return b[n/2];
}
```

label	command	arguments
`median:`	`push`	`EBP`
	`mov`	`EBP, ESP`

label	command	arguments

موفق باشید!