TA: Jade Cheng

ICS 312

Homework Solution #5

Due Date: September 10, 2009

Exercise #10.1

Question: If we have defined

```
Msg       db          'Hello, friends!', 13, 10, '$'
```

What is displayed by

Answer:

```
_PutStr     Msg           ; 'Hello, friends!'
_PutStr     Msg + 7       ; 'friends!'
mov         Msg, 'M'
mov         Msg + 5, 'w'
_PutStr     Msg           ; 'Mellow friends!'
```

Exercise #10.2

Question: Figure out what each of the following code fragments does. You can trace each fragment using the data definitions in Exercise 1, but you should give a general description of what is being accomplished, not what is happening to the particular data above.

a.

```
            sub      ax, ax
            mov      bx, 10
Mystery1:
            add      ax, [WArray + bx]
            sub      bx, 2
            jge      Mystery1
```

Answer: This program loops from the last element of `WArray` to the first element of this array, computes the summation of the elements, and stores it in the register `ax`.

b.

```
            mov      ax, 1
            mov      bx, 0
```

```
                mov     cs, 6
Mystery3:
                mov     [WArray + bx], ax
                add     bx, 2
                inc     ax
                dec     cx
                jnz     Mystery3
```

Answer: This program loops from the beginning of `WArray` to the end of this array, and re assign every element from 1 to 6. The `WArray` array becomes 1, 2, 3, 4, 5, 6 and the end of the execution.

c.

```
                mov     bx, 0
                mov     cx, 11
Mystery4:
                cmp     [WArray + bx], ‘0’
                jl      M4
                cmp     [CArray + bx], ‘9’
                jle     Done
M4:             inc     bx
                dec     cx
                jnz     Mystery4
Done:
```

Answer: This program loops from the beginning of `CArray` until it reaches ‘8’ in this array, and it terminates.

Question: Rewrite each of the mystery programs in Exercise 2 using the pointer method (`[bx]`) instead of the subscript method (`[WArray + bx]` or `[CArray + bx]` indexing).

a.

```
                sub     ax, ax
                mov     bx, 10
Mystery1:
                add     ax, [WArray + bx]
                sub     bx, 2
                jge     Mystery1
```

Answer:

```
                sub     ax, ax
                mov     bx, offset WArray + 10
Mystery1:
                add     ax, [bx]
                sub     bx, 2
                jge     Mystery1
```

b.

```
             mov      ax, 1
             mov      bx, 0
             mov      cs, 6
Mystery3:
             mov      [WArray + bx], ax
             add      bx, 2
             inc      ax
             dec      cx
             jnz      Mystery3
```

Answer:

```
             mov      ax, 1
             mov      bx, offset WArray
             mov      cs, 6
Mystery3:
             mov      [bx], ax
             add      bx, 2
             inc      ax
             dec      cx
             jnz      Mystery3
```

c.

```
             mov      bx, 0
             mov      cx, 11
Mystery4:
             cmp      [WArray + bx], ‘0’
             jl       M4
             cmp      [CArray + bx], ‘9’
             jle      Done
M4:          inc      bx
             dec      cx
             jnz      Mystery4
Done:
```

Answer:

```
             mov      bx, offset CArray
             mov      cx, 11
Mystery4:
             cmp      [bx], ‘0’
             jl       M4
             cmp      [bx], ‘9’
             jle      Done
M4:          inc      bx
             dec      cx
             jnz      Mystery4
Done:
```

Question: Suppose that A, B and C are arrays of 100 words.

a. Write code to place in `ax` the smallest integer in A. Use the pointer method (`[bx]` indexing) as opposed to the subscript method.

Answer: *The pointer method:*

```
               mov      bx, offset A + 2
               mov      ax, [A]
               mov      cx, 100
  Program:
               cmp      ax, [bx]
               jle      Increment
               mov      ax, [bx]
  Increment:
               add      bx, 2
               dec      cx
               jnz      Program
```

The subscript method:

```
               mov      bx, offset A + 2
               mov      ax, [A]
               mov      cx, 100
  Program:
               cmp      ax, [bx]
               jle      Increment
               mov      ax, [bx]
  Increment:
               add      bx, 2
               dec      cx
               jnz      Program
```

b. Write code to set all entries in A to 1. Use the pointer method.

Answer: *The pointer method:*

```
               mov      bx, offset A
               mov      cx, 100
  Program:
               mov      [bx], 1
               add      bx, 2
               dec      cx
               jnz      Program
```

Question: Suppose that we have recorded the tab stops we wish to use in byte array TabStop, with the property that TabStop[0]< TabStop[1] <···< TabStop[N], which is the highest set tab stop (N is a word variable). If COL is the byte variable containing the current position of the cursor and a tab is entered, the new value of COL is the smallest TabStop[I] which is greater than COL. If no such value exists, the new value of COL is COL + 1. Write assembler code to compute this new value of COL.

Answer:

```
                  mov      cx, N
                  mov      bx, offset TabStop
    Program:
                  cmp      [bx], COL
                  jg       Found
                  inc      bx
                  dec      cx
                  jnz      Program
                  inc      COL
                  jmp      Done
    Found:        mov      COL, [bx]
    Done:
```