2. Consider the following pseudocode.

Initialize a DWORD in memory to 0

while (ECX ≠ 0) {

Read a nonnegative integer from the keyboard into EAX

Increase the value stored in memory by adding the value from EAX

Decrease the value of ECX by 1

}

Display the value stored in memory

Add labels and jump instructions to the following code to implement the while loop.

.data

saved DWORD 0

.code

top: ;top of loop label

jecxz bottom ;exit statement

; Loop body: read from keyboard, add to memory, decrement ECX

call ReadDec

add saved, eax

sub ecx, 1

jmp top ;repeat statement

bottom: ;end of loop label

; Display value stored in memory

mov eax, saved

call WriteDec

3. Storing the doubleword 6A7B8B6Ah requires four bytes of memory. How would these bytes be

laid out in memory on a computer that uses

1. big endian byte ordering?

6A 7B 8B 6A

1. little endian?

6A 8B 7B 6A

4. Consider the following .data section.

.data

a SWORD 10, 10h, –3

WORD 0, 2 DUP(3)

DWORD 83829FC0h

1. How many bytes of memory are allocated to store this data?

16

(b) How will this data be stored in memory as a sequence of bytes? Write the byte values in

hexadecimal, starting from the byte at the lowest memory address.

0A 10 FD 00 03 03 C0 9F 82 83

5. For each of the following data declarations, determine if the data declaration is valid or invalid.

(Try to figure it out yourself first – this is better preparation for exams – then use MASM to check.)

If the declaration is invalid, (a) write the error message that MASM produces, then (b) describe, in

your own words, what is wrong with the declaration. If there is an obvious way to correct the error,

write the corrected declaration.

(a) nums WORD 10 20 30 40 ; Array of four words

(b) BYTE ?

(c) BYTE 256

(d) WORD 'x'

(e) WORD "Hello",0

(f) twofiftyfive WORD FFh ; Hexadecimal FF

(g) ebp BYTE "ebp",0 ; Null-terminated string ebp

(h) empty DWORD 4\*1024 DUP(?)