Part 2 – Computer Science 112 – Midterm Review – Spring 2018

3.Implement each design given below.

IMPORTANT be able to implement AND and OR conditions

Design: Assume value is in EDX

if (value <= -1000) or (value >= 1000)

then

value: = 0

end if;

cmp edx, -1000 ; value <= -1000 ?

jle thend ; do it if so

cmp edx, 1000 ; value >= 1000 ?

jnge endifd ; skip if not

thend: mov edx, 0 ; value := 0

endifd:

1. Implement each design given below. Assume that *value* is stored as a doubleword in memory, *number* is in EAX, and *count* is in ECX.

IMPORTANT be able to implement AND and OR conditions

while (count < 100) or (value ≤ 500) loop

add count to value;

add 2 to count;

end while;

whileC:

cmp ecx, 100 ; count < 100?

jl bodyC ; do body if so

cmp value, 500 ; value <= 500?

jnle endwhileC ; exit if not

bodyC:

add value, ecx ; value + count

add ecx, 2 ; count + 2

jmp whileC ; repeat

endwhileC:

1. Implement each design given below. Assume that *value* is stored as a doubleword in memory, *number* is in EAX, and *count* is in ECX.

IMPORTANT be able to implement AND and OR conditions

count := 0;

value := 50;

repeat

add 1 to count;

add count to value;

until count > 10 or value ≥ 150;

mov ecx, 0 ; count := 0

mov value, 50 ; value := 50

untilC:

inc ecx ; add 1 to count

add value, ecx ; add count to value

cmp ecx, 10 ; count > 10

jg enduntilC ; exit if so

cmp value, 150 ; value >= 150?

jnge untilC ; repeat if not

enduntilC:

6.This problem has a for loop implemented with a “loop” statement. How many times is the loop body executed?

mov ecx, 7

forC

……. Loop body

Answer: 7

7.Exercise 5.5 Page 161

No. 1- Modify the program in Figure 5.9 replacing the second loop by one that changes each number larger than average to zero.

In general be able to write code to process an array of integers

...

idiv nbrElts ; calculate average

; change each array element above average to 0

lea ebx,nbrArray ; get address of nbrArray

mov ecx,nbrElts ; count := nbrElts

forCount2:

cmp [ebx],eax ; number > average ?

jng endIfLarge ; continue if not greater

mov DWORD PTR [ebx], 0 ; change number to 0

endIfLarge:

add ebx,4 ; get address of next item of array

loop forCount2 ; repeat

quit: INVOKE ExitProcess, ; exit with return code 0

8.

This problem contains a design with a for loop. Assume that sum reference a doubleword in the data segment. Give a fragment of 80x86 code that implements the design . Use a loop statement appropriately in the code.

sum = 0;

for count := 1 to 50 loop

add count to sum

end for;

mov sum, 0 ; sum := 0

mov ecx, 50 ; loop index

forB: mov eax, 51 ; count = 51 – index

sub eax, ecx

add sum, eax ; add count to sum

loop forB ; repeat

Understand the mod-reg-r/m byte: Be able to recognize it and get the 8 bits and know what the three parts are in general

Bits 7 and 6 - 2 bit Mod “mode” field

Bits 5, 4, 3 – 3 bit Reg for register field

Bits 2, 1, 0 – 3 bit R/M Register/Memory field

Microsoft (R) Macro Assembler Version 14.00.24210.0

03/29/18 00:54:29

homework1.asm Page 1 – 1

; Homework 1mov eax, 100

.586

.MODEL FLAT

.STACK 4096

00000000 .DATA

00000000 FFFFFFFF dValue DWORD -1

00000004 00000000 dValueF DWORD ?

00000000 .CODE

00000000 main PROC

00000000 BB 0000FF75 mov ebx, 0000FF75h

00000005 B9 000001A2 mov ecx, 000001A2h

0000000A 8B D9 mov ebx, ecx (a)

1. *mod-reg-r/m* byte D9 = 1101 1001 = 11 011 001, *mod* 11 for register-to-register, *reg* 011 for EBX and *r/m* 001 for ECX

0000000C B8 000001A2 mov eax, 000001A2h

00000011 B8 00000064 mov eax, 100 (b)

1. no *mod-reg-r/m* byte

Please note that 00000064 is the hex value of the immediate 100

00000016 BA FF754C2E mov edx,0FF754C2Eh

0000001B 8B 15 00000000 R mov edx, dValue (c)

1. *mod-reg-r/m* byte 15 = 0001 0101 = 00 010 101, *mod* 00 and *r/m* 101 for direct memory operand, *reg* 010 for EDX

00000021 66| B8 014B mov ax, 014Bh

00000025 B4 00 mov ah, 0 (d)

1. no *mod-reg-r/m* byte

Please note 00 is the hex value of 0 the immediate value

00000027 B0 64 mov al,64h

00000029 B0 FF mov al, -1 (e)

1. no *mod-reg-r/m* byte

Please note that FF is the hex value of -1 the immediate value

0000002B BB 00003A4C mov ebx, 00003A4Ch

00000030 89 1D 00000004 R mov dValueF, ebx (f)

*mod-reg-r/m* byte 1D = 0001 1101 = 00 011 101, *mod* 00 and *r/m* 101 for direct memory operand, *reg* 011 for EBX

00000036 B9 00000000 mov ecx, 00000000h

0000003B B9 00000080 mov ecx, 128 (g)

no *mod-reg-r/m* byte Please note that 00000080 is the hex value of 128 the immediate value

00000040 B8 00000000 mov eax, 0 ; exit with return code 0

00000045 C3 ret

00000046 main ENDP

END