CPSC 240: Computer Organization and Assembly Language
Assignment 08, Spring Semester 2023

CWID: 886880327 Name: Andrew Saldana

1. Download the "CPSC-240 Assignment08.docx" document.
2. Design the "macro.asm" program, input a value n (n=100 ~ 255) from the keyboard, calculate 1+2+3+…+n, and display the calculation result in the terminal emulator window. The corresponding C/C++ code is as follows:

```
#begin define print(string, numOfChar)
     rax = 1;
     rdi = 1;
     rsi = &string;
     rdx = numOfChar;
     syscall;
#end
#begin define scan(buffer, numOfChar)
     rax = 1;
     rdi = 1;
     rsi = &buffer;
     rdx = numOfChar;
     syscall;
#end

char buffer[4];
long n;
short sumN;
char msg1[26] = "Input a number (100~255): ";
char msg2[16] = "1 + 2 + 3 +...+ ";
char msg3[4] = " = ";
char ascii[6] = "00000\n";

print(msg1, 26);
scan(buffer, 4);
n = atoi(buffer);
rsi = 0;
do {
    sumN += rsi;
} while(rsi >= 0);
ascii = itoa(sumN);
print(msg2, 20);
print(buffer, 3);
print(msg3, 3);
print(ascii, 6);
```

3. Run the "macro" file to display the calculation result in the Terminal Emulator window.
4. Insert source code (macro.asm) and simulation results (Terminal Emulator window) at the bottom of the document. Write an analysis to verify the simulation results.
5. Save the file in pdf format and submit the pdf file to Canvas before 23:59 pm on 04/30/2023.

Simulation Sample:

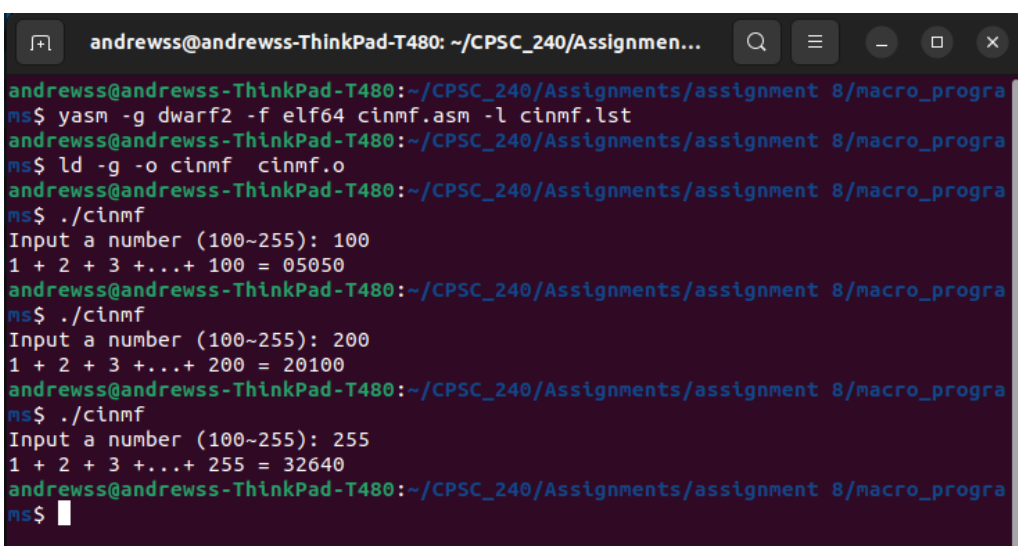
```
899486336@vclvm011308-225-78: ~/Desktop/ex8
File  Edit  View  Search  Terminal  Help
899486336@vclvm011308-225-78:~/Desktop/ex8$ yasm -g dwarf2 -f elf64 ex8.asm
899486336@vclvm011308-225-78:~/Desktop/ex8$ ld -g -o ex8 ex8.o
899486336@vclvm011308-225-78:~/Desktop/ex8$ ./ex8
Input a number (100~255): 100
1 + 2 + 3 +...+ 100 = 05050
899486336@vclvm011308-225-78:~/Desktop/ex8$ ./ex8
Input a number (100~255): 200
1 + 2 + 3 +...+ 200 = 20100
899486336@vclvm011308-225-78:~/Desktop/ex8$ ./ex8
Input a number (100~255): 255
1 + 2 + 3 +...+ 255 = 32640
899486336@vclvm011308-225-78:~/Desktop/ex8$
```

[Insert macro.asm source code here]

```
%macro  print   2
        mov     rax, 1                          ;SYS_write
        mov     rdi, 1                          ;standard output device
        mov     rsi, %1                         ;output string address
        mov     rdx, %2                         ;number of character
        syscall                                 ;calling system services
%endmacro

%macro  scan    2
        mov     rax, 0                          ;SYS_read
        mov     rdi, 0                          ;standard input device
        mov     rsi, %1                         ;input buffer address
        mov     rdx, %2                         ;number of character
        syscall                                 ;calling system services
%endmacro

section .bss
buffer          resb    4
n               resq    1
sumN            resw    1

section .data
msg1            db      "Input a number (100~255): "
msg2            db      "1 + 2 + 3 +...+"
msg3            db      " = "
ascii           db      "00000", 10

section .text
        global _start
_start:
;-----------------------------------------------------------------------------------------------------------
; When user input is given, it is represented in ascii so the following code gets every ascii
; digit that was user inputed and indivdually changes them into a decimal number. We prompted
; the user to enter a three digit numbr so we will be converted asccii into decimal three different times
;
;
; HOW IT WORKS(for example we have '111' as user input): changes the first char input into decimal and saves that into
; al:ah(1), multiplies that that number by 10 (10), changes the second char input into decimal and adds
; that to al:ah(11), multiplies by 10 again (110), changes the third char input into decimal and adds that
; to al:ah(111), and then moves that whole number into n.

        print   msg1, 26                        ;cout << msg1
        scan    buffer, 4                       ;cin >> buffer

        ;printing msg2, buffer, and msg3 before we modify buffer
        print   msg2, 16                        ;cout << msg2
        print   buffer, 3                       ;cout << ascii_input
        print   msg3, 3                         ;cout << msg3

```

```
        mov     ax, 0                                   ;clear ax
        mov     bx, 10                                  ;bx = 10
        mov     rsi, 0                                  ;counter = 0
next0:
        and     byte[buffer+rsi], 0fh                   ;convert ascii to number
        add     al, byte[buffer+rsi]                    ;al = number
        adc     ah, 0                                   ;ah = 0
        cmp     rsi, 2                                  ;compare rsi with 2
        je      skip0                                   ;if rsi=2 goto skip0
        mul     bx                                      ;dx:ax = ax * bx
skip0:
        inc     rsi                                     ;rsi++
        cmp     rsi, 3                                  ;compare rsi with 3
        jl      next0                                   ;if rsi<3 goto next0
        mov     word[n], ax                             ;n = ax
;-----------------------------------------------------------------------------------------------------------------------
        ; calculates 1+2+3+...+N
        ;if user inputs 140-255, the

        mov     cx, 0                                   ;cx = 0
next1:
        add     word[sumN], cx                          ;sumN += cx
        inc     cx                                      ;cx++
        cmp     cx, word[n]                             ;compare cx with n
        jbe     next1                                   ;if(cx<=100) goto next1
;-----------------------------------------------------------------------------------------------------------------------
        ; converts sumN into ascii
        ; Depending on the inputed number, sumN may be 4 or 5 digits. Therefore, we will set up ascii to hold 5 digits.
        ; But if sumN ends up being 4 digits, ascii will add an extra 0 to the beginning of the number

        mov     rdi, 4                                  ;counter = 4
        mov     ax, word[sumN]                          ;ax = sumN
next2:
        mov     dx, 0                                   ;dx = 0
        mov     bx, 10                                  ;bx = 10
        div     bx                                      ;dx=(dx:ax)%10, ax=(dx:ax)/10
        add     byte[ascii+rdi], dl                     ;ascii+rdi = al + 30h
        dec     rdi                                     ;rdx--
        cmp     rdi, 0                                  ;compare rdx with 0
        jge     next2                                   ;if rd  x>=0 goto next2

;-----------------------------------------------------------------------------------------------------------------------
        ; printing results
        print   ascii, 6                                ;cout << ascii



        mov     rax, 60                                 ;terminate program
        mov     rdi, 0                                  ;exit status
        syscall                                         ;calling system services
```

[Insert macro simulation result here]



[Insert macro simulation analysis here]

User input ascii → decimal

buffer= '123'

rsi	convert buffer[rsi] to decimal	al = al+buffer[rsi]	rsi = 2?	inc rsi	al x10
0	1	al = 1	no, continue	rsi = 1	al=10
1	2	al = 12	no, continue	rsi = 2	al=120
2	3	al=123	yes, stop		

calculates 1+2+3+... + N

n=123

CX	SumN = CX+SumN	CX++	CX ≤ 123?
0	SumN = 0	CX = 1	yes, jmp
1	SumN = 1	CX = 2	yes, jmp
2	SumN = 3	CX = 3	yes, jmp
3	SumN = 6	CX = 4	yes, jmp
↓	↓	↓	↓
123	SumN = 7626	CX = 124	No, stop

Sum ⇒ asci

Sum = 7626

```
ax=762
10|7626
  7626
  ----
dx= 6
↓
36h
```

```
ax=76
10|762
  760
  ---
dx= 2
↓
32h
```

```
ax=7
10|76
  70
  --
dx= 6
↓
36h
```

```
ax= 0
10|7
  0
  -
dx= 7
↓
37h
```

asci = 37h, 36h, 32h, 36h = '7626'

macros

```
%macro print 2
    mov rax, 1
    mov rdi, 1
    mov rsi, %1
    mov rdx, %2
    syscall
%endmacro
```

```
%macro scan 2
    mov rax, 0
    mov rdi, 0
    mov rsi, %1
    mov rdx, %2
    syscall
%endmacro
```

26 characters

msg1 = "Input a number (100-255): "

print msg1, 26

↓

```
mov rax, 1
mov rdi, 1
mov rsi, msg1
mov rdx, 26
syscall
```

4 characters

buffer = '123 ↵'

Scan buffer, 4

↓

```
mov rax, 0
mov rdi, 0
mov rsi, buffer
mov rdx, 4
syscall
```