nasm -f elf -o 3.o 3.s
gcc -no-pie -m32 3.o -o 3
./3

1.add two nmbrs & str in c

```
extern     printf
extern     scanf

SECTION .data
a:    dd    0
b:    dd    0
c:    dd    0

enter:     db "Enter two numbers: ",0
out_fmt:   db "%ld + %ld =%ld", 10, 0
out_fmt_2: db "%s",10,0
in_fmt:          db "%d",0

SECTION .text

global main
main:
        PUSH out_fmt_2
        PUSH enter
        call printf
        add esp, 8

      PUSH a
      PUSH in_fmt
      call scanf
      add esp,8

      PUSH b
      PUSH in_fmt
      call scanf
      add esp,8

      mov   eax,[a]
      mov   ebx,[b]
      add   eax,ebx
      mov   [c],eax

      mov eax, [a]
      mov ebx, [b]
```

```
mov ecx, [c]

PUSH ecx
PUSH ebx
PUSH eax
PUSH out_fmt
   call    printf
   add esp, 16

mov   eax,0
ret
```

2.Scan three variables a,b, and c. Print the value of 2a + 3b + c

3.sum from 1 to x. You may assume x is a positive integer.

```
extern printf
extern scanf

SECTION .data
x:      dd 0
sum:    dd 0
msg1:   db "Enter a positive integer: ",0
out_fmt: db "Sum from 1 to %d = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
   ; Ask input
   push msg1
   call printf
   add esp,4

   push x
   push in_fmt
   call scanf
   add esp,8

   mov ecx,[x]
   mov eax,0
```

```
.loop:
    cmp ecx,0
    jle .done
    add eax,ecx
    dec ecx
    jmp .loop

.done:
    mov [sum],eax

    push dword [sum]
    push dword [x]
    push out_fmt
    call printf
    add esp,12

    mov eax,0
    ret
```

4.GCD

```
extern printf
extern scanf

SECTION .data
a: dd 0
b: dd 0
msg: db "Enter two numbers: ",0
out_fmt: db "GCD = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
    push msg
    call printf
    add esp,4

    push a
    push in_fmt
    call scanf
```

```
    add esp,8

    push b
    push in_fmt
    call scanf
    add esp,8

    mov eax,[a]
    mov ebx,[b]

.gcd:
    cmp ebx,0
    je .done
    mov edx,0
    div ebx
    mov eax,ebx
    mov ebx,edx
    jmp .gcd

.done:
    push eax
    push out_fmt
    call printf
    add esp,8

    mov eax,0
    ret
```

5. prime or not.

```
extern printf
extern scanf

SECTION .data
x: dd 0
msg: db "Enter a number: ",0
prime_msg: db "%d is prime",10,0
not_prime: db "%d is not prime",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
```

```
    push msg
    call printf
    add esp,4

    push x
    push in_fmt
    call scanf
    add esp,8

    mov eax,[x]
    cmp eax,2
    jl .notprime

    mov ecx,2
.loop:
    mov edx,0
    mov ebx,ecx
    div ebx
    cmp edx,0
    je .notprime
    inc ecx
    mov eax,[x]
    cmp ecx,eax
    jl .loop

    push dword [x]
    push prime_msg
    call printf
    add esp,8
    jmp .end

.notprime:
    push dword [x]
    push not_prime
    call printf
    add esp,8

.end:
    mov eax,0
    ret
```

6. maximum of the three numbers.

```
extern printf
extern scanf

SECTION .data
a: dd 0
b: dd 0
c: dd 0
msg: db "Enter three numbers: ",0
out_fmt: db "Maximum = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
    push msg
    call printf
    add esp,4

    push a
    push in_fmt
    call scanf
    add esp,8

    push b
    push in_fmt
    call scanf
    add esp,8

    push c
    push in_fmt
    call scanf
    add esp,8

    mov eax,[a]
    mov ebx,[b]
    mov ecx,[c]

    cmp ebx,eax
    jle .skip1
    mov eax,ebx
.skip1:
    cmp ecx,eax
    jle .skip2
    mov eax,ecx
```

```
.skip2:

    push eax
    push out_fmt
    call printf
    add esp,8

    mov eax,0
    ret
```

7.reverse of the number. (Example: 123 → 321)

```
extern printf
extern scanf

SECTION .data
n: dd 0
rev: dd 0
msg: db "Enter a number: ",0
out_fmt: db "Reverse = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
    push msg
    call printf
    add esp,4

    push n
    push in_fmt
    call scanf
    add esp,8

    mov eax,[n]
    mov ebx,0      ; rev = 0

.loop:
    cmp eax,0
    je .done
    mov edx,0
    mov ecx,10
    div ecx
```

```
    imul ebx,ebx,10
    add ebx,edx
    jmp .loop

.done:
    mov [rev],ebx
    push dword [rev]
    push out_fmt
    call printf
    add esp,8

    mov eax,0
    ret
```

8. multiplication table of n (from 1 to 10)

```
extern printf
extern scanf

SECTION .data
n: dd 0
msg: db "Enter a number: ",0
out_fmt: db "%d x %d = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
    push msg
    call printf
    add esp,4

    push n
    push in_fmt
    call scanf
    add esp,8

    mov ecx,1
.loop:
    cmp ecx,11
    jge .done
    mov eax,[n]
    imul eax,ecx
```

```
    push eax
    push ecx
    push dword [n]
    push out_fmt
    call printf
    add esp,16

    inc ecx
    jmp .loop

.done:
    mov eax,0
    ret
```

9.Print all divisors of n.

```
extern printf
extern scanf

SECTION .data
n: dd 0
msg: db "Enter a number: ",0
out_fmt: db "%d ",0
newline: db 10,0
in_fmt: db "%d",0

SECTION .text
global main
main:
    push msg
    call printf
    add esp,4

    push n
    push in_fmt
    call scanf
    add esp,8

    mov eax,[n]
    mov ebx,1

.loop:
```

```
    cmp ebx,[n]
    jg .done
    mov edx,0
    div ebx
    cmp edx,0
    jne .next

    push ebx
    push out_fmt
    call printf
    add esp,8

.next:
    inc ebx
    mov eax,[n]
    jmp .loop

.done:
    push newline
    call printf
    add esp,4

    mov eax,0
    ret
```

10. Declare an array to hold 20 integers. Repeatedly read 20 integers from the user (one at a time). For each integer, add it to a running sum. Store the integers in the array. After all 20 integers have been entered, print the total sum of the numbers. Finally, print back all the numbers stored in the array in the order they were entered.

```
extern printf
extern scanf

SECTION .data
arr: times 20 dd 0
msg1: db "Enter 20 integers:",10,0
out_sum: db "Sum = %d",10,0
out_all: db "You entered:",10,0
in_fmt: db "%d",0
out_fmt: db "%d ",0
newline: db 10,0

SECTION .bss
sum resd 1
```

```
SECTION .text
global main
main:
    push msg1
    call printf
    add esp,4

    mov ecx,0
    mov dword [sum],0

.loop_input:
    cmp ecx,20
    jge .done_input

    lea eax,[arr+ecx*4]
    push eax
    push in_fmt
    call scanf
    add esp,8

    mov eax,[arr+ecx*4]
    add [sum],eax

    inc ecx
    jmp .loop_input

.done_input:
    push dword [sum]
    push out_sum
    call printf
    add esp,8

    push out_all
    call printf
    add esp,4

    mov ecx,0
.loop_print:
    cmp ecx,20
    jge .end
    push dword [arr+ecx*4]
    push out_fmt
    call printf
```

```
    add esp,8
    inc ecx
    jmp .loop_print

.end:
    push newline
    call printf
    add esp,4
    mov eax,0
    ret
```

11. palindrome

```
extern printf
extern scanf

SECTION .data
str: times 100 db 0
msg: db "Enter a string: ",0
yes_msg: db "%s is a palindrome",10,0
no_msg:  db "%s is not a palindrome",10,0
in_fmt:  db "%s",0

SECTION .text
global main
main:
    ; Ask for input
    push msg
    call printf
    add esp,4

    ; Read string
    push str
    push in_fmt
    call scanf
    add esp,8

    ; Set pointers
    mov esi, str      ; start
    mov edi, str

find_end:
    cmp byte [edi],0
    je got_end
    inc edi
```

```
    jmp find_end

got_end:
    dec edi          ; edi now at last character

check_loop:
    cmp esi,edi
    jge palindrome     ; done checking, all matched
    mov al,[esi]
    mov bl,[edi]
    cmp al,bl
    jne not_palindrome
    inc esi
    dec edi
    jmp check_loop

palindrome:
    push str
    push yes_msg
    call printf
    add esp,8
    jmp done

not_palindrome:
    push str
    push no_msg
    call printf
    add esp,8

done:
    mov eax,0
    ret
```

12. takes two integers as input, passes them to a function sum, and prints their sum.

```
extern printf
extern scanf

SECTION .data
a: dd 0
b: dd 0
msg: db "Enter two integers: ",0
out_fmt: db "Sum = %d",10,0
in_fmt: db "%d",0
```

```
SECTION .text
global main
extern sum

main:
    push msg
    call printf
    add esp,4

    push a
    push in_fmt
    call scanf
    add esp,8

    push b
    push in_fmt
    call scanf
    add esp,8

    push dword [b]
    push dword [a]
    call sum
    add esp,8

    push eax
    push out_fmt
    call printf
    add esp,8

    mov eax,0
    ret

sum:
    push ebp
    mov ebp,esp
    mov eax,[ebp+8]
    add eax,[ebp+12]
    pop ebp
    ret
```

13. function max_of_two, and prints the larger number.

```
extern printf
```

```
extern scanf

SECTION .data
a: dd 0
b: dd 0
msg: db "Enter two integers: ",0
out_fmt: db "Max = %d",10,0
in_fmt: db "%d",0

SECTION .text
global main
extern max_of_two

main:
    push msg
    call printf
    add esp,4

    push a
    push in_fmt
    call scanf
    add esp,8

    push b
    push in_fmt
    call scanf
    add esp,8

    push dword [b]
    push dword [a]
    call max_of_two
    add esp,8

    push eax
    push out_fmt
    call printf
    add esp,8

    mov eax,0
    ret

max_of_two:
    push ebp
    mov ebp,esp
```

```
    mov eax,[ebp+8]
    mov ebx,[ebp+12]
    cmp eax,ebx
    jge .done
    mov eax,ebx
.done:
    pop ebp
    ret
```

14. reads a string, reverses it using a function reverse_str, and prints the reversed string.

```
extern printf
extern scanf

SECTION .data
str: times 100 db 0
msg: db "Enter a string: ",0
out_fmt: db "Reversed: %s",10,0
in_fmt: db "%s",0

SECTION .text
global main
extern reverse_str

main:
    push msg
    call printf
    add esp,4

    push str
    push in_fmt
    call scanf
    add esp,8

    push str
    call reverse_str
    add esp,4

    push str
    push out_fmt
    call printf
    add esp,8
```

```
    mov eax,0
    ret

;----------------------------------
; reverse_str(char *s)
reverse_str:
    push ebp
    mov ebp,esp
    mov esi,[ebp+8]    ; s
    mov edi,esi
.find_end:
    cmp byte [edi],0
    je .got_end
    inc edi
    jmp .find_end
.got_end:
    dec edi
.swap:
    cmp esi,edi
    jge .done
    mov al,[esi]
    mov bl,[edi]
    mov [esi],bl
    mov [edi],al
    inc esi
    dec edi
    jmp .swap
.done:
    pop ebp
    ret
```

15. "and" on two matrices of the same size

16. Write a NASM program that takes a student's name (string), three exam scores (integers), and computes: (i)The average score (float), (ii) The corresponding letter grade (P/F), and prints a formatted output line. Above 50% marks is P.