- **01**

```
section .rodata
   msg db "This is my first try to use assembler and see the
execution",10,0
   msg1 db "%d",0
   msg2 db "VAlue is %d",10,0
;;; scanf("%d",&a);
;;; printf("Value is %d", a);
;;; & =address *=Value at

section .bss
   a resd 1
section .text
   global main
   extern printf,scanf

main:
   push msg
   call printf
   add esp,4

   push a
   push msg1
   call scanf
   add esp,8

   push dword[a]
   push msg2
   call printf
   add esp,8
```

- **02**

```
section .rodata
```

```
 msg db  "This is my first assembly code!",10,0
 msg1 db  "%d",0
 msg2 db  "Value is %d",10,0

section .bss
 a resd 1

section .text
 global main
 extern printf,scanf

main:
push msg
 call printf
add esp,4

push a
push msg1
 call scanf
add esp,8
 push dword[a]
push msg2
 call printf
add esp,8
```

- **03**

```
   section .text
   global main
main:
   mov ecx,edx
   mov eax,ecx
   mov eax,ebx
   mov cx,dx
   mov ax,cx
   mov ax,bx

   add ecx,edx
   sub eax,ecx
```

```
    cmp eax,ebx
    add cx,dx
    sub ax,cx
    cmp ax,bx
    xchg ecx,edx
    xchg eax,eax        ;0
    xchg eax,ecx        ;1
    xchg eax,edx        ;2
    xchg eax,ebx        ;3
    xchg eax,esp        ;4
    xchg eax,ebp        ;5
    xchg eax,esi        ;6
    xchg eax,edi        ;7
```

- **04**

```
section .text
    global main
main:
    inc eax
    inc ecx
    inc edx
    inc ebx
    inc esp
    inc ebp
    inc esi
    inc edi
    inc al
    inc ah
    inc bl
    inc bh
    inc ax

    dec eax
    dec ecx
    dec edx
    dec ebx
    dec esp
    dec ebp
```

```
   dec esi
   dec edi
   dec al
   dec ax
```

- **05**

```
section .text
   global main
main:
   mov eax,10
   mov ecx,10
   mov edx,10
   mov ebx,10
   mov esp,10
   mov ebp,10
   mov esi,10
   mov edi,10

   mov ax,10
   mov cx,10
   mov dx,10
   mov bx,10
   mov sp,10
   mov bp,10
   mov si,10
   mov di,10
```

- **06**

```
section .text
   global main
main:
   add eax,10
   add ecx,10
   add edx,10
   add ebx,10
   add esp,10
   add ebp,10
   add esi,10
   add edi,10

   add eax,1234
   add ecx,1234
   add edx,1234
   add ebx,1234
   add esp,1234
   add ebp,1234
   add esi,1234
   add edi,1234

   add al,12

   add ax,1234
   add cx,1234
   add dx,1234











   add ax,10
   add cx,10
   add dx,10
   add bx,10
   add sp,10
   add bp,10
   add si,10
   add di,10
```

- **07**

```
section .text
   global main
main:
   add eax,ebx
   add ebx,eax
   cmp eax,ebx
   cmp ebx,eax
   sub eax,ebx
   sub ebx,eax

   add eax,123456
   add ebx,123456
   cmp eax,123456
   cmp ebx,123456
   sub eax,123456
   sub ebx,123456










   xchg eax,ebx
   xchg ebx,ecx
   xchg eax,eax
   xchg ebx,ebx
   nop
   xchg ebx,eax
   add eax,10
   add eax,12345
   add eax,123456789

   add ebx,10
   add ebx,12345
   add ebx,123456789
   mov eax,10
   mov eax,12345
   mov eax,123456789
   mov ebx,10
   mov ebx,12345
   mov ebx,123456789
```

- **08**

```
section .bss
   sum resd 1
   sum1 resw 1
   sum2 resb 1
section .text
   global main
main:
   mov [eax],edx
   mov dword[esp],10
   mov dword[ebp],10
   mov dword[esp],eax
   mov dword[esp],ebx

   mov eax,10
   mov ax,10
   mov al,10
   mov ah,10
   mov ebx,10
   mov bx,10
   mov bl,10
   mov bh,10
   mov dword[sum],10
   mov word[sum1],10
   mov byte[sum2],10
   mov dword[esp],10
   mov dword[edx],10
   mov dword[sum],esi
   mov dword[sum],eax

   ;; mov word[ax],10
   ;; mov byte[al],10
```

- **09**

```
section .data
   Array dd 110 , 220 , 330 , 440 , 550 , -1
   msg db "Addition is %d",10,0

section .bss
   sum resd 1
   Four resd 1
section .text
   global main
   extern printf
main:
   mov dword[Four],4
   mov dword[sum],0
   xor  ecx,ecx        ; ecx= 0
;;; Address Element : Array + Index * size
;;; 10 : 100 + 0 * 4 = 100
;;; 20 : 100 + 1 * 4 = 104
;;; 30 : 100 + 2 * 4 = 108
lp: mov esi, Array
   mov eax, 4
   ;;  mov edi, dword[esi+ ecx * eax]
   mul ecx          ; eax = eax * ecx
   add esi, eax
   mov edi, dword[esi+ecx* 4]
   mov edi, dword[Array+ecx* 4]

   cmp edi,-1
   jz print
   add dword[sum],edi
   ;; add dword[sum],dword[esi]
   inc ecx
   jmp lp

print:
   push dword[sum]
   push msg
   call printf
   add esp,8

```

- **10**

```
   section .bss
   a resd 1
   section .text
   global main
main:
   mov eax, dword[a +10]
   mov edx, dword[a+12345]
;;; mod 01
   mov eax, dword[edx +10]
   mov edx, dword[ecx+100]
;;; mod 10
   mov eax, dword[esi +12345]
   mov edx, dword[edi+12345]

   add eax, dword[a +10]
   add edx, dword[a+12345]
;;; mod 01
   add eax, dword[edx +10]
   add edx, dword[ecx+100]
;;; mod 10
   add eax, dword[esi +12345]
   add edx, dword[edi+12345]

```

- **11**

```
   section .bss
   base resd 1
   index resd 1
section .text
```

```
    global main
main:

    mov edx , [ebp+eax]
    mov edx , [ebp+ecx]
    mov edx , [ebp+eax*2]
    mov edx , [ebp+ecx*2]
    mov edx , [esp+eax]
    mov edx , [esp+ecx]
    mov edx , [esp+eax*2]
    mov edx , [esp+ecx*2]
    mov edx , [esp+eax*4]
    mov edx , [esp+ecx*4]
    mov edx , [esp+eax*8]
    mov edx , [esp+ecx*8]


    mov dword[index],4
    mov dword[base],eax
    mov eax, [ebx+ecx]
    add eax, [ebx+ecx]
    mov eax, [esi+edi]
    mov eax, [base+eax]
    mov eax, [base+ecx]
    mov eax,base
```

- **12**

```
    section .text
    global _start
_start:
    mov eax,1
    mov ebx,42
    int 0x80
```

- **13**

```
    section .text
    global _start
```

```
_start:
   xor eax,eax
   inc eax
   mov ebx,42
   int 0x80

```

- **14**

```
   section .text
   global main
main:
   mov eax,42
   ret

```

- **15**

```
   section .data
   msg db "This is just and example to show how we can get a small
size executable",10,0
   len equ $-msg
   section .text
   global main
main:
   mov eax,4
   mov ebx,1
   mov ecx,msg
   mov edx,len
   int 0x80

   mov eax,1
   mov ebx,0
   int 0x80

```

- **16**

```
   section .text
   global main
main:
   mov eax,10
   mov ecx,20
   jmp abc      ; unconditional forward jump
   jmp abc
   jmp abc
   jmp abc
   jmp abc
   jmp abc
   add eax,23
pqr:
   sub ecx,19
abc:
   dec eax
   dec ebx
   jmp pqr      ; unconditional Backward jump
   jmp pqr
   jmp pqr
```

- **17**

```
   section .text
   global main
main:
pqr:mov eax,10      ; 5 bytes
   mov ecx,20       ; 5 bytes
   add eax,23       ; 3 bytes
   sub ecx,19       ; 3 bytes
   jz abc           ; 2 bytes; 2+2+2+2+2+1+1+2+2+2 = 18 = 12
   jz abc           ; 2 bytes; 2+2+2+2+1+1+2+2+2 = 16 = 10
   jz abc           ; 2 bytes; 2+2+2+1+1+2+2+2 = 14 = 0E
   jz abc           ; 2 bytes; 2+2+1+1+2+2+2 = 12 = 0C
   jz abc           ; 2 bytes; 2+1+1+2+2+2 = 10 = OA
   jz abc           ; 2 bytes; 1+1+2+2+2=08 = 08
   dec eax          ; 1 byte
   dec ebx          ; 1 bytes
   jz pqr           ; 2 bytes ; 2+1+1+2+2+2+2+2+2+3+3+5+5 = 32 =
0010 0000 = 1101 1111 = 1110 0000 E0
   jz pqr           ; 2 bytes; 2+2+1+1+2+2+2+2+2+2+3+3+5+5 = 34 =
0010 0010 = 1101 1101 = 1101 1110 DE
```

```
   jz pqr          ; 2 bytes; 2+2+2+1+1+2+2+2+2+2+2+3+3+5+5 = 36 =
0010 0100 = 1101 1011 = 1101 1100 DC
abc:
```

- **18**

```
   section .text
   global main
main:
pqr:mov eax,10      ; 5 bytes
   mov ecx,20       ; 5 bytes
   add eax,23       ; 3 bytes
   sub ecx,19       ; 3 bytes
   jmp abc          ; 2 bytes; 2+2+2+2+2+1+1+2+2+2 = 18 = 12
   jmp abc          ; 2 bytes; 2+2+2+2+1+1+2+2+2 = 16 = 10
   jmp abc          ; 2 bytes; 2+2+2+1+1+2+2+2 = 14 = 0E
   jmp abc          ; 2 bytes; 2+2+1+1+2+2+2 = 12 = 0C
   jmp abc          ; 2 bytes; 2+1+1+2+2+2 = 10 = OA
   jmp abc          ; 2 bytes; 1+1+2+2+2 = 08 = 08
   dec eax          ; 1 byte
   dec ebx          ; 1 bytes
   jmp pqr          ; 2 bytes; 2+1+1+2+2+2+2+2+2+3+3+5+5 = 32 =
0010 0000 = 1101 1111 = 1110 0000 E0
   jmp pqr          ; 2 bytes; 2+2+1+1+2+2+2+2+2+2+3+3+5+5 = 34 =
0010 0010 = 1101 1101 = 1101 1110 DE
   jmp pqr          ; 2 bytes; 2+2+2+1+1+2+2+2+2+2+2+3+3+5+5 = 36 =
0010 0100 = 1101 1011 = 1101 1100 DC
abc:
```