M. Daniyal Rana.   L1F20BSCS0036

COAL

Assignment 2

Q1

Add two 64-bit no. using e.a.t.

```
    [org 0x100]
    xor ax, ax
    xor bx, bx
    xor cx, cx
    xor dx, dx

    mov ax, [a]
    mov bx, [a+2]
    mov cx, [a+4]
    mov dx, [a+6]

    add ax, word [b]
    add bx, word [b+2]
    add cx, word [b+4]
    add dx, word [b+6]

    mov ax, 0x4C00
    int 21h
a: dq 0x1234567812345678
b: dq 0x1245897643123918
```

Q2

Rotation of 64 bit number, 6 times

```
[org 0x100]

    xor ax, ax
    xor bx, bx
    xor cx, cx
    xor dx, dx

    mov ax, [a]
    mov bx, [a+2]
    mov cx, [a+4]
    mov dx, [a+6]

start:

    shr ax, 1
    rcr bx, 1
    rcr cx, 1
    rcr dx, 1

mov word [a], ax
mov word [a+2], bx
mov word [a+4], cx
mov word [a+6], dx

counter:
 cmp byte [count], 6
 jne inc
 jmp end
```

```
inc:
add byte [count], 1
jump start

end:
mov ax, 0x4c00
int 21h

a: dq 500cH

count: db 0x1
```

Q3

Multiply a 32 bit number

```
[org 0x100]
xor ax,ax
xor bx,bx
xor cx,cx
xor dx,dx

mov bx , [mult]
mov cx , [mult+2]
mov dx , [mult +4]
bit check:
Shr dx,1
rcr cx,1
rcr bx,1
jnc skip
mov ax, [multc]
add [result+0],ax
mov ax , [multc +2]
adc [result +2] , ax
mov ax, [multc+4]
adc [result+4], ax

adc
skip:
shl word [multc],1
rcl word [multc+2],1
rcl word [multc+4],1
dec byte [count]
jnz bit check
mov ax,0x4c00
int 21h
```

```
multc : dd 23045
mult : dd 1000
result : dq 0
count : db 32.
```

Q4

```
[org 0x100]
 xor  ax,ax
 xor  bx, bx
 xor  cx, cx
 xor  dx,dx
 mov  ax, [num]
 mov  dx,0
 mov  bx,0
 jmp  L1
 Label 1
    inc  dx
    add  bx, 2
    cmp  bx, 0xFFFF
    jmp  Skip 2
 L1 :
 cmp  ax , [cs:bx]
    JL  Label 1
   add  bx, 2
  cmp  bx, 0xFFFF
    jnz  L1
    jmp  Skip 2
L2:
    inc dx
  add bx,2
 cmp  bx, 0xFFFF
   jz  Skip 3
   jnz  Label 2
```

```
Skip2:
  mov  bx, 0
Label2:
 cmp  ax, [ds:bx]
   jL  L2
  add  bx, 2
 cmp  bx,0xFFFF
   jnz  Label 2
  jmp  skip 3
L3:
   cmp  ax, [ss:bx]
    jL  L3
  add  bx,2
 cmp  bx, 0xFFFF
   jnz  Label 3
   Skip 4:
  mov  ax ,0x4c00
   int  21 h.
```

Scanned with CamScanner

Q5

```
[org 0x100]
 xor   ax, ax
 xor   bx, bx
 xor   cx, cx
 xor   dx, dx
          →   mov ax, 5
 Push ax      mov bx, 7
 Push bx      mov cx, 9
mov BP, SP    mov bp, 0
PUSH  cx      mov sp, 0xFFFE
PUSH  BP
PUSH  ax
POP  dx
mov  ax, [BP+2]
Add  ax, [BP-4]
POP ax
Add  SP, 8
PUSH AX

 mov  ax, 0x4c00
 int  21 h
```

Final values:

AX = FFFA
BP = FFFA
DX = 0005
SP = FFFE
CX = 0009