Experiment No: 08-08-24

ODD OR EVEN

Aim:- Implement a ODD or Even program in MASM programming

Algorithm

1. START
2. Define data segment
3. Initialize msg1 "Enter the number: $" msg 2 The number is Even: $", msg 3 "The number is odd: $"
4. End of data segment
5. Code segment
6. Assume cs:code ds:data
7. Load the data segment into Ax
8. Move Ax into Ds
9. Load the address of the prompt message into DX
10. Display the message 1
11. Input a character
12. int 21h to read the input
13. Adjust the input character from ASCII to its numeric
14. Rotate right through carry to check the least significant bit
15. Jump to odd label if Carry is set
16. Load the address of msg 2
17. Display msg 2
18. Jump to stop, the end of the program
19. Load the address of msg 3
20. Display msg 3
21. mov ah, 4ch function to exit the program
22. Terminate program

23. code ends
24. Stop

Result : MASM program for odd or even executed successfully and output obtained.

28/10/24

PROGRAM

```
data segment
m1 db 0ah,0dh,"enter the number:$"
m2 db 0ah,0dh,"the number is even:$"
m3 db 0ah,0dh,"the number is odd:$"
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
lea dx,m1
mov ah,09h
int 21h
mov ah,01h
int 21h
add al,48
rcr al,1
jc odd
lea dx,m2
mov ah,09h
int 21h
jmp stop
odd:lea dx,m3
mov ah,09h
int 21h
stop:mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

```
enter the number: 4
the number is even
enter the number: 3
the number is odd
```

Experiment No :-

21-08-24

16-BIT Addition

Aim: Implement 16-bit addition of two numbers in MASM programming.

Algorithm

1. START
2. Define data segment
3. Initialize msg1 "Enter first number: $", msg2 "Enter second number $" msg3 "Result $"
4. Define n_1 and n_2 7 byte
5. End of data segment
6. Define macro display with parameter msg
7. Used to display msg
8. Define macro read Digit
9. Used to read data from user and convert Ascii value to numeric value.
10. Define macro print Digit
11. Convert numeric value to Ascii and print data in DL
12. Define code segment.
13. Assume cs: code ds: data
14. Initialize label start
15. Point ds register to beginning of data segment.
16. Initialize si and di with offset of n_1 and n_2.
17. Display msg1, move value 04h to cl for counter.
18. Initialize label first
19. Read 16 bit digit one by one to address specified in si, decrement cl
20. Read until value of cl not equal to zero.

21. Initialize label second read value of second 16 digit number same as first by incrementing di.
22. Initialize label addition.
23. Add first and second number one by one decrement di, si and cl
24. Add until value of cl not zero.
25. Display the result printDigit
26. Terminate program
27. Code segment ends
28. Stop

Result : MASM program for 16-bit addition executed successfully and output obtained.

25/10/24

PROGRAM

```
data segment
msg1 db 0ah,0dh,"first no:$"
msg2 db 0ah,0dh,"second no:$"
msg3 db 0ah,0dh,"result:$"
n1 db 07h dup(?)
n2 db 07h dup(?)
data ends
display macro msg
lea dx,msg
mov ah,09h
int 21h
endm
readDigit macro
mov ah,01h
int 21h
sub al,30h
endm
printDigit macro
add dl,30h
mov ah,02h
int 21h
endm
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov si,offset n1
mov di,offset n2
display msg1
mov cl,04h
first:
readDigit
mov [si],al
inc si
dec cl
jnz first
display msg2
mov cl,04h
```

```
second:
readDigit
mov [di],al
inc di
dec cl

jnz second
clc
mov cl,04h
addition:
dec di
dec si
mov al,[si]
mov bl,[di]
adc al,bl
mov ah,00h
aaa
mov [di],al
dec cl
jnz addition
display msg3
mov cl,04h
print:
mov dl,[di]
printDigit
inc di
dec cl
jnz print
mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

first no: 1121
second no: 3214
result: 4335

Experiment no:- 21-08-24

16-BIT Subtraction

Aim: Implement 16-bit subtraction of two numbers in MASM programming.

Algorithm

1. START
2. Define data segment
3. Intialize msg1 "Enter first number: $", msg2 "Enter second number $" msg3 "Result $"
4. Define n1 and n2 7 byte
5. End of data segment
6. Define macro display with parameter msg
7. Used to display msg
8. Define macro readDigit
9. Used to read data from user and convert ASCII value to numeric value.
10. Define macro printDigit
11. Convert numeric value to ASCII and print data in DL
12. Define code segment
13. Assume cs:code ds:data
14. Initialize label start
15. Point ds register to beginning of data segment.
16. Initialise si and di with offset of n1 and n2.
17. Display msg1, move value 04h to cl for counter
18. Initialize label first
19. Read 16 bit digit one by one to address specified in si, increment si decrement cl
20. Read until value of cl not equal to zero.

21. Initialize label second read value of second 16 digit number same as first by incrementing di
22. Initialize label subtraction
23. Subtract first and second number one by one decrement di, si and cl
24. Subtract until value of cl not zero.
25. Display the result printDigit
26. Terminate program
27. Code segment ends
28. Stop

<u>Result</u>:- MASM program for 16-bit subtraction executed successfully and the output obtained.

28/10/24

PROGRAM

```
data segment
msg1 db 0ah,0dh,"first no:$"
msg2 db 0ah,0dh,"second no$"
msg3 db 0ah,0dh,"result:$"
n1 db 07h dup(?)
n2 db 07h dup(?)
data ends
display macro msg
lea dx,msg
mov ah,09h
int 21h
endm
readDigit macro
mov ah,01h
int 21h
sub al,30h
endm
printdigit macro
add dl,30h
int 21h
endm
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov si,offset n1
mov di,offset n2
display msg1
mov cx,04h
first:
readDigit
mov[si],al
inc si
dec cx
jnz first
display msg2
mov cx,04h
second:
readDigit
mov[di],al
```

```
inc di
dec cx
jnz second

clc
mov cx,04h
subtraction:
dec di
dec si
mov al,[si]
mov bl,[di]
sbb al,bl
mov ah,00h
aas
mov [di],al
dec cx
jnz subtraction
display msg3
mov cx,04h
print:
mov dl,[di]
printDigit
inc di
dec cx
jnz print
mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

first no: 9999
second no: 4444
result: 5555

Experiment No:- 01-02-24

32-BIT Addition

Aim:- Implement 32 bit addition of two numbers in MASM programming.

Algorithm

1. START
2. Define data segment
3. Initialize msg1 "Enter first number : $", msg2 "Enter second number $" msg3 "Result $"
4. Define n1 and n2 a byte
5. End of data segment
6. Define macro display with parameter msg
7. Used to display msg
8. Define macro readDigit
9. Used to read data from user and convert ASCII value to numeric value.
10. Define macro printDigit
11. Convert numeric value to ASCII and print data in DL.
12. Define code segment.
13. Assume cs:code ds:data
14. Initialize label start
15. Point ds register to beginning of data segment
16. Initialize si and di with offset of n1 and n2.
17. Display msg1, move value 08h to cl for counter
18. Initialize label first
19. Read 32 bit digit one by one to address specified in si increment si decrement cl
20. Read until value of cl not equal to zero

21. Initialize label second read value of second 32 bit number same as first by incrementing value of di
22. Initialize label addition
23. Add first and second number one by one decrement di, si, cl
24. add until value of cl not zero.
25. Display result printDigit
26. Terminate Program
27. Code segment ends
28. Stop

Result : MASM program for 32 bit addition executed successfully and output obtained.

28/10/24

PROGRAM

```
data segment
msg1 db 0ah,0dh,"enter the first number:$"
msg2 db 0ah,0dh,"enter the second number:$"
msg3 db 0ah,0dh,"result:$"
n1 db 09h dup(?)
n2 db 09h dup(?)
data ends
display macro msg
lea dx,msg
mov ah,09h
int 21h
endm
readDigit macro
mov ah,01h
int 21h
sub al,30h
endm
printDigit macro
add dl,30h
mov ah,02h
int 21h
endm
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov si,offset n1
mov di,offset n2
display msg1
mov cx,08h
first:
readDigit
mov[si],al
inc si
dec cx
jnz first
display msg2
mov cx,08h
second:
readDigit
```

```
mov[di],al

inc di
dec cx
jnz second
clc
mov cx,08h
addition:
dec di
dec si
mov al,[si]
mov bl,[di]
adc al,bl
mov ah,00h
aaa
mov[di],al
dec cx
jnz addition
display msg3
mov cx,08h
print:
mov dl,[di]
printDigit
inc di
dec cx
jnz print
mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

enter the first number: 50000000
enter the second number: 10000000
result: 60000000

Experiment No:- 4 21-08-24

32 - BIT Subtraction

Aim:- Implement subtraction of two 32 bit numbers in MASM programming.

Algorithm

1. START
2. Define data segment
3. Initialize msg1 "Enter first number $", msg2 "Enter Second number $" msg3 "Result $"
4. Define n1 and n2 a byte
5. End of data segment
6. Define macro display with parameter msg
7. Used to display msg
8. Define macro readDigit
9. Used to read data from user and convert ASCII value to numeric value.
10. Define macro printDigit
11. Convert numeric value to ASCII and print data in DL
12. Define code segment
13. Assume cs: code, ds: data
14. Intialize label start
15. Point ds register to beginning of data segment
16. Intialize si and di with offset of n1 and n2.
17. Display msg1, move value 08h to cl for counter.
18. Intialize label first
19. Read 32 bit digit one by one to address specified in si, increment si decrement cl
20. Read until value of cl not equal to zero.

21. Intialize label second read value of second 32 bit number same as first by incrementing di
22. Intialize label subtraction
23. Subtract first and second number one by one decrement di, si and cl
24. Subtract until value of cl not zero
25. Display the result print Digit
26. Terminate program
27. Code segment ends
28. Stop

<u>Result</u>: MASM program for 32 bit subtraction executed successfully and output obtained.

28/10/24

PROGRAM

```
data segment
msg1 db 0ah,0dh," first number:$"
msg2 db 0ah,0dh," second number:$"
msg3 db 0ah,0dh,"result:$"
n1 db 09h dup(?)
n2 db 09h dup(?)
data ends
display macro msg
lea dx,msg
mov ah,09h
int 21h
endm
readDigit macro
mov ah,01h
int 21h
sub al,30h
endm
printDigit macro
add dl,30h
mov ah,02h
int 21h
endm
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov si,offset n1
mov di,offset n2
display msg1
mov cx,08h
first:
readDigit
mov[si],al
inc si
dec cx
jnz first
display msg2
mov cx,08h
second:
readDigit
```

```
mov[di],al

inc di
dec cx
jnz second
clc
mov cx,08h
subtraction:
dec di
dec si
mov al,[si]
mov bl,[di]
sbb al,bl
mov ah,00h
aas
mov[di],al
dec cx
jnz subtraction
display msg3
mov cx,08h
print:
mov dl,[di]
printDigit
inc di
dec cx
jnz print
mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

first number: 30000000
second number: 10000000
result: 20000000

Experiment No: 21-02-24

Linear Search

Aim: Implement linear search in MASM programming

Algorithm

1. START
2. Define data segment
3. Intialize prompt 1 "Enter the string $", prompt 2 "Enter the key $", result 1 "key Found $", result 2 "key not found $"
4. Define array 9 byte
5. End of data segment
6. Define macro display with parameter msg
7. Used to display msg
8. Define readcharacter macro
9. Used to read character ASCII value
10. Define code segment
11. Assume cs:code, ds:data
12. Initialize label start
13. point ds register to beginning of data segment
14. display prompt 1, cl:00h
15. Initialize label stringscam
16. Read character until AL equal to 0dh
17. move value in al to address in si increment si & cl
18. Intialize label ended
19. Display prompt 2.
20. read character store value in BL
21. Intialize label check
22. Compare all characters with character in BL

23. If found jump to found
24. Else jump to not found.
25. Intialize label found.
26. display result 1 jump to finish.
27. Intialize label not found
28. Display result 2
29. Intialize label finish
30. terminate program
31. Code segment ends
32. Stop

Result:- MASM program to implement linear search executed successfully and output obtained.

28/10/24

PROGRAM

```
data segment
m1 db 0ah,0dh,"enter the string:$"
m2 db 0ah,0dh,"enter the key:$"
r1 db 0ah,0dh,"key found $"
r2 db 0ah,0dh,"key not found$"
array db 09h dup(?)
data ends
display macro msg
lea dx,msg
mov ah,09h
int 21h
endm

readCharacter macro
mov ah,01h
int 21h
endm
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov si,offset array
display m1
mov cl,00h
stringScan:
readCharacter
cmp al,0dh
jz ended
mov [si],al
inc cl
inc si
jmp stringScan
ended:
display m2
readCharacter
mov bl,al
mov ch,00h
check:
dec si
```

```
cmp bl,[si]
jz found

dec cl
jnz check
jmp notfound
found:
display r1
jmp finish
notfound:
display r2
finish:
mov ah,4ch
int 21h
code ends
end start
```

OUTPUT:

```
enter the string: locker
enter the key: o
key found
enter the string: lol
enter the key: c
key not found
```