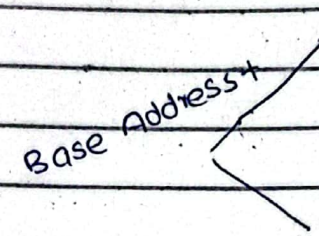
Microprocessor & Assembly Language 29th Nov, 2021

Statistics & Probability Theory

-> Arrays:

List of elements of same type.

```
MSG     DB      "Hello World"
NUB     DW      10, 30, 40
```



NUB	100h	10
NUB+2h	102h	30
MSG	100h	H
MSG+1h	101h	e
MSG+2h	102h	l

```
Five    DB    5 DUP(0)
```

↑ common values.

Duplicate:

Defined arrays whose index share common values.

SWAP

```
MOV  AX, NUB+18h.
XCHG AX, NUB+48h
MOV  NUB+18h, AX.
```

For DW Array A. of 10 elements.

```
MOV  Ax, 0.
MOV  CX, 10.
MOV  Dx, 0
MOV  SUM, 0
Repeat:
     Add  AX, A+Dx
     ADD  DX, 2
     LOOP repeat.
```

Micro processor & Assembly Language

-> Indirect Address Mode
way to specify the operands.

Register Indirect Mode [register]

ADD AX, [SI]

BX, SI, ~~BP~~, DI, BP → SS : BP

DS SS

DS : BX

DS : SI

DS : DI

↓ ↘ offset address.

~~stack~~ segment word

For Extra segment
MOV AX, ES:[SI]

-> Direct Address Mode

1) register mode
ADD, AX, BX

2) Immediate mode
ADD AX, 7

3) Direct mode
ADD AX, A

Data segment

MOV BX, 1000h		
MOV SI, 2000h	1000h	1BACh
MOV DI, 3000h		
	2000h	2BCh
MOV BX, [BX] // ✓	1BACh	
MOV CX, [SI] ✓	3000h	031Bh
MOV CX, [AX] X		
MOV [BI], [SI] X		

[LEA SI, A]
MOV CX, SI

INC[DI] ✓

→

A DW 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

```
XOR AX, AX
MOV CX, 10
LEA SI, A.
Repeat:
    ADD AX, [SI]
    ADD SI, 2
LOOP Repeat;
```

A	10
A+2	20
A+4	30
⋮	⋮

→ Based & Indexed Addressing Modes.

BX, BP
SI, DI

[Register + Displacement]
[Displacement + register]
[Register] + Displacement
Displacement + [Register]
Displacement [Register]

MOV [BP], 0.
MOV [BP+2], 0.

Displacement may be
* A variable
* A constant (Negative / Positive)
* offset ± constant

if Register is BX / BP then
Based Addressing mode

If register is SI / DI then
Indexed addressing mode

convert to upper case.

```
DB 'this is byte'
MOV CX, 12
LEA SI, MSG.
Repeat
CMP [SI], ' '
JE exit
SUB
ADD [SI], 20h.
ADD SI, 1.
Exit
ADD SI, 1
loop repeat.
MOV DX, SI
MOV AH, 9
INT 21h
```

A	10
A+2	20
A+4	30
A+6	40

```
MOV BX, 4
MOV AX, A[B
MOV AX, [A+B
MOV AX, [BX+
MOV AX, A+[B
MOV AX, [BX]
```

```
XOR SI, SI
MOV CX, 12
Repeat:
CMP MSG[SI], ' '
JE Next
SUB MSG[SI], 20h
Next:
INC SI
loop Repeat.
```

Page-4

-> PTR

```
A DB ,8
MOV AX, A      X.
MOV AX, WORD PTR A.
```

BYTE
Byte
DWORD

-> 2D Arrays:-

```
A   DB   10,20,30
    DB   40,50,60
    DB   70,80,90,
```

Row Major Order

Column Major Order

A
10
20
30
40
50
:
90

10
40
70
20
50
80
:
90

for column wise

$A[i,j] = A + ((i-1) + (j-1)*M)*S$

Row — M = 3 | Column — N = 3 | size — S = 1

$A[i,j] = A + ((i-1)*N + (j-1))*S$

for Row wise

Based Indexed Addressing Mode

The offset address of operand is sum

i) Content of Based Register (Bx, BP)
ii) Content of index Register (DI, SI)
iii) Optimally, a variable offset Address
iv) Optimally, a constant

The operand may be written as

i) Variable [Base-Reg] [Index Reg]
ii) [Base-Reg + Index-Reg + variable + constant]
iii) Variable [Base_Reg + Index-Reg + constant]
iv) constant [Base-Reg + Index-Reg + variable]

```
MOV  BX, 2
MOV  SI, 4
MOV  AX, A[BX][SI]
MOV  AX, [A+BX+SI]
MOV  AX, A[BX+SI]
```

LEA AX, A

Selection Sort :-

```
i = N
For N-1 times DO
    Find the position K of the largest
    element among A[1] .... A[i]
    Swap A[K] & A[i]
        i = i - 1
end for.
```

Microprocessor & Assembly Language

12th Dec, 2023

P1:-

Write down Reverse procedure that will reverse the array with N items SI pointing to the array and BX contains the N.

∴

If we SHL, 1 it multiplies by 2.

If we SHR, 1 it devides by 2.

P2:-

Selection sort.

i = N

For N-1 times Do

Find the position K of largest element among A[1] ..... A[i]

swap A[K] & A[i]

i = i-1.

end for

P1:-

```
A DW 1, 2, 3, 4, 5, 6, 78
LEA SI, A          ; Offset of array.
MOV DI, SI         ; points to 1st index.
MOV CX, BX         ; array size
SHR CX, 1          ; 7/2 => 3
DEC BX             ; 6
SHL BX, 1          ; 12.
ADD DI, BX         ; 0+12 => A+12.
Repeat:
MOV AX, [SI]       ; get 1st element
XCHG AX, [DI]      ; exchange with last elem
MOV [SI], AX       ; store in 1st place
```

```
ADD SI, 2 ;  points to 2nd element
SUB DI, 2 ;  points to 2nd last ||
loop Repeat

MOV cx, 7.
LEA SI, A
Y2:
MOV DX, [SI]
ADD DL, 30h
MOV AH, 2
int 21h
ADD SI, 2
loop Y2
```

X ——— X ——— X .

```
A DW   10, 20, 30, 40
   DW   50, 60, 70, 80
   DW   90, 100, 110, 120.
```

clear 2nd Row.

```
MOV Bx, 8
XOR SI, SI
MOV cx, 4
Repeat:
   MOV A[Bx][SI], 0
   ADD SI, 2
loop Repeat.
```

10.6 :-

	Test1	Test2	Test3
Ali	10	8	3
Ahmed	12	7	5
Khan	8	6	7
Qasim	9	5	10

A DW

10, 8, 3

12, 7, 5

8, 6, 7

9, 5, 10