Imp. Points about AL.

1- Variable —— memory operand

2- array —— multiple initialization

3- function —— procedure, subroutine

4- INC, DEC does not affect carry flag

5- NEG Instruction to nonzero operand alway set CF

6- Procedure TextEQU <PROC> (EQU: symbolic constant | symbol definition)

7- Current location counter $, DUP

8- Overflow — same sign operands

9- jitna size memory operand — utna register

10- C++ array initialize with loop — dup()

11- index —— offset

12- WriteString — cout<< (edx) → write null-terminated string to standard output.

mWriteStr calls this procedure

↳ macro

msg Byte "Enter name:", 0

mov edx, offset msg

call WriteString

return args

↓ none

(bzw) data accumulator registor

address / distance beginning to enclosing segment

Date:

M T W T F S

(13) ReadInt Proc

32-bit signed decimal integer, stop → Enter key

Before non-numeric character, all valid ~~integer~~ character converts to integer

Leading space ignored, leading +,- permitted

if value != 32-bit signed int → reset eax, set overflow flag and display error msg

call args none

Return args

OF=0, EAX= valid binary val and SF=sign

OF=0, EAX=0 (invalid input)

call ReadInt ——— cin >> eax

Unsigned Int — ReadDec procedure

hexadecimal — ReadHex procedure

(14) WriteInt Proc

write signed 32-bit decimal to standard output in decimal format with a leading sign and no leading zeros.

return args: none

Call args: EAX = signed number to write

```
mov eax, 216543
call writeInt
```

——— output — +216543

Unsigned Integer — WriteDec procedure

hexadecimal — WriteHex

Binary — writeBin

mShow - macro

Date: _____ M T W T F S

15) direct (var)

direct-offset (var+2)

indirect (WORD PTR [esi])

indexed (var[esi])

base-indexed (word ptr [ebx+esi])

base-indexed-displacement ([var+ecx+edi])

16) mShow vari format → optional

mshow dval default (HIN)

hexadecimal, signed dec, newline

mShow MACRO itsName:REQ, format:=<HIN>

17. String Input:-

msg Byte 10 dup(?)

.code

mov edx, offset msg
mov ecx, sizeof msg
call readString
call writeString

18 int 21h — getch()

19 .data? — uninitialized data seg

20. option casemap: none — make case sensitive except a few identifiers

Date:

M T W T F S

Imp. Points about AL.

21- Jmp → EIP — unconditional target within same procedure usually

22- Label → Block {} cpp

23. Write Char — al registey used.

24. one ~~string~~ character + integer can be added and displayed using write char

→ mov al, '3'
mov al, '4'
call writechar

; mov eax, '3' → '34'
; mov eax, '4'

last char will be added
4+4 ⇒ 8

→ if sum range exceed from one char limit, it will prints some other unusual characters/symbols

'9' + 3 ⇒
':' + 3 , '2' + 35

→ when both chars are added, 2nd value will always be the result.

'3' + '5' ⇒ 5

25- space → 20h , 32 decimal

26. Line Feed → Ah , 10 decimal

27- Carriage Return → Dh , 13 decimal

28- ExitProcess PROTO , QWORD , eax → rax (64 bit registor) R64

64 bit

call ExitProcess → END → END main

Date:_____ M T W T F S

29- 64bit loop counter → RCX reg as counter
32 + 64 bit mode loopd use → ECX register.
32+64bit mode loopw use → CX reg as counter.

30- x86 Processors use Little-Endian Order to store values in memory, (RAM). Values will be stored in reverse order like:-

12345678h ⇒ (5678, 1234)
dword → (word →

31- writeString → edx (parameter)
writeInt, writeDEC, writeHex, writeBin ⇒ eax, numeric value
writeChar → al, extended key → al = 0, ah = keyboard scan code

32- By default value decimal. 10d = 10

33- mul ebx ⇒ imul (optional) eax, ebx

34- Runtime stack → ESP, downward, higher to lower address
decrement (by stack element size)

→ Individual Procedure Description

1- call waitMsg — getch(), Enter key

2- call crlf, clrscr, delay → eax → 1000 = 1sec

3- DumpMem → hexadecimal loop, esi starting address (offset arr)
ecx → length of arr, ebx → type 3 params

loop → decimal, int, string — for all purposes

Date: _____ M T W T F S

4- DumpRegs → no input or return value, cpu snapshot, debugging — 10 regs, 6 flags

5- GetMseconds — return EAX (milliseconds) elapsed since mid night

6- GotoXY — X - coordinate (column) 0-79 DL
Y - coordinate (row) 0-24 DH

7- IsDigit — input AL → char ⇒ if 0-9 ZF = 1 (set)
else ZF = 0
mov AL, 'a'
call IsDigit
↳ zero flag = 0 (clear)

8- Random32 — return EAX — simple function having input (seed) put in a formula to generate number
subsequent value used previously generated random value as their seeds.

9- Randomize — initialize starting seed value for Random32, RandomRange
Seed = time of day, accurate to 1/100 of a second
Each time generate unique value.

10- RandomRange — 0 - n-1, input and return ⇒ EAX

11- ReadHex — EAX input, return — last chars taken only 8
1234567898765321 | space, commas, special chars = 0
1,2,3,4 ⇒ 01020304 ← writeHex ← writeDec

12- ReadString — EDX offset, ECX max num of chars size of
Enter stop, add terminating null byte
return EAX, length of only chars except 10

13- SETtextColor — input EAX ⇒ white + (blue * 16) → bg color
to upperline 16 underline ↔ shift left ↔ (blue SHL 4)

Date: _____ M T W T F S

Imp... Points about AL.

14- Length of — counts null character also

15- Write Char — mov al, 65 ASCII to char conversion

16- Write Hex — 8 digits output, input, output return any

35- AND al, 11011111 ⇒ converting characters into uppercase
(AND: ↓ intersection; 11011111: ↓ reverse)

Bitwise inst — clear overflow, carry — sign, zero, parity modify

36- NOT ⇒ complement, no flag set -, not reg, not mem

37- OR ⇒ Union

38- Cmp destination, source — compare int ✓ float ✗
implied subtraction of src from dest op
no op modified, logical (boolean) exp

39- TEST, AND with 0 ⇒ Set zero flag
- OR an operand with 1 ⇒ clear zero flag
- TEST does not modify operand, AND does
- high OR with 1 ⇒ set sign flag
- (highest bit AND with 0 ⇒ clear sign flag
- STC ⇒ set carry flag
- CLC ⇒ clear carry flag
- add 2 positive values, produce negative sum ⇒ set overflow flag
- OR an operand with 0 ⇒ clear overflow flag

Date: _____ M T W T F S

40- **64-bit** if src less than 32 bit and dst 64 bit

all bits are affected

- src = 32 bit, dst = ~~64~~ bit

only lower 32 bit of dst are affected

```
mov rax, FFFFFFFF FFFF FFFFh
AND rax, 80808080h
```

RAX ⇒ FFFF FFFF 80808080

```
mov rax, FFFF FFFF FFFF FFFF
AND rax, 80808080h
```

RAX ⇒ 0000 0000 0080 8080

41- OR al, 00100000b ⇒ convert char into low case