1/6/17

System Software

* compilers - convert HL language to LL lang

[Assembly lang- much more ~~system~~ Hardware oriented
It can be understand by hardware programmer
only. More space needed

* assembler

* Interpreters - line by line checking

* Linker

Simple c program.

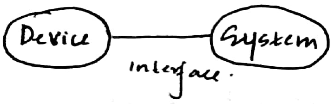
```
#include<stdio.h>
main()
{  int a, b, c;
     a=1
     b=2
     c=a+b;
     Printf ("c=" %d); }
```

header files are used to recognize the function

Linker: links the function used in the headerfile
with the program function. Here printf
is a function included in stdio.h

*loader- loads to main m/y and Executed.

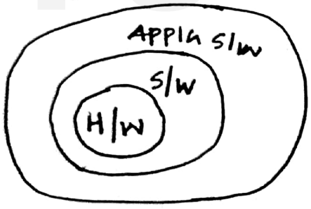
*device driver - It is a software



- machine dependancy
- OS specific.

```
#include <stdio.h>
# define ~~max~~ MAX 100    (macro)-replaces
                                 keyword.
main()
 {
    MAX }
```

* OS- interface b/w user and system
  - manages hardware and software.



System software	Application software.
• m/c dependant	• m/c independent.
• interact with h/w directly	• Interacts with h/w indirectly through s/m calls.
• Collection of programs enable users to interact with h/w	• Collection of Programs Written for a specific application.
• coding is complex	• coding is easy
• compilers, interpreter, OS.	• Chrome Ms-word, Ms-pain

✓ SIC - Simplified Instructn component

✓ SIC/XE (Extra equipment)

upward ~~compactablity~~ combatible

SIC

An object prgm for the std SIC m/c it also execute properly on SIC/XE System

MEMORY :-

8 bit byte

3 bytes = 1 word (24 bits)

Byte addresses:-

$2^{15}B = 32768$

Registers:-

Small and faster m/y storage. In SIC we we use 5 Registers

A	0	Accumulator	S-4
X	1	Index Register.	~~X~~ T-5
L	2	Linkage	
✓ PC	8	Program counter	
SW	9	Status Word	

Data formats:-

- Integers stored as 24 bit number.
- -ve stored as 2's complement.
- char is represented as 8 bit ASCII code.

Instruction formats:- (24-bit)

8	1	15
opcode	x	address

↓ 0/1 (index)

Addressing modes:

Depending on the value of x we can choose the mode.

	Mode	Indication	Target address
1.	Direct	x=0	TA = address
2.	Ind~~irec~~ext	x=1	TA = add +(X)

Instruction Set:-

LDA, LDX
STA, STX } Load and store.

ADD, SUB
MUL, DIV } arithmetic op

COMP
JLT, JEQ, JGT } conditional

~~FH~~ JSUB
RSUB } Jump & return to Subroutine.

I/O:-

Transfers one bit at a time from right mos
8 bits of A register & each devices assigned an
8 bit code. There are 3 o/p ~~&~~ i/p instructn

- TD - test device.
  RD Read drive.
  WD Write drive.

SIC / XE

Memory → 8 bit bytes.

3 B = 1 word.

byte address - 2^{20} B = 1 MB

Registers

B	3	Base register
S	4	GP
T	5	GP
F	6	Floating ~~place~~ point acc (48 bits)

Data format.

Floating pt

1	11	36
S	Exponent	fraction

sign bit

Instruction purpose

Format 1 (1 B)

8
opcode

Format 2 (2 B)

8	4	4
opcode	r_1	r_2

Format 3 (3B)

6	1	1	1	1	1	1	12
opcode	n	i	x	b	p	e	disp

Format 4 (4 B)

6	1	1	1	1	1	1	20
opcode	n	i	x	b	p	e	address

n- Indirect
i- immediate
x- index
b- base-relative
p- PC_relative
e- format 3/4 will be choosen.

e-Expone
e-extended

- Base_relative: b=1, P=0, TA = (B) + displacemen
- PC_ relative: b=0, P=1, TA = (PC)+ displacement
- Imm. address: i=1, n=0, TA = Value at ~~word~~ location
- Indirect " : i=0, n=1, TA - Value at word
- Simple addressing mode: i=0, n=0, TA = locatn of operand
  or
  i=1, n=1
- Indexed x=1, TA = (X) + address
- Direct x = 0, TA = address

Q:1 Find target address and value loaded into accumulator from the given hex value.

A) $x = 032600$ (convert to binary)

θ — 0000 00|1|1| 0|0|1|0| 0110 0000 0000

n i x b p e

$n=1,\ i=1,\ x=0,\ b=0,\ P=1,\ e=0$

Simple address

PC-relative addressing mode

[This is also given in qstn]

	⋮
3030	003600
	⋮
3600	103000
	⋮
6390	006303
	⋮
C303	003030

B = 006000

PC = 003000

X = 000090

TA = (PC) + displacement

= 003000 + 600 = 003600//

Value in 3600 is 103000. This value is loaded to accumulator.

Q:2 Find 03C300

A) 0 000 00|1|1| 1|1|0|0| 0011 0000 0000

$n=1,\ i=1\ \ x=1,\ b=1,\ P=0,\ e=0$

- Simple address
- Base relative ($b=1,\ P=0$)
- Indexed $x=1$

Target address = (B) + (x) + displacement.

= 006000 + 000090 + 300

= 6390.

Q)3 022030

0000 00|1|0|0|0|1|0| 0000 0011 0000

$h=1,\ i=0,\ x=0,\ b=0,\ P=1,\ e=0$

- Indirected
- PC-relative.

TA = (PC) + displacement.

= 003000 + 030 = 003030. = 3030

TA = Value at word = 103000

Q:4 010030

Q:5 003600

Q:6 0310C303

A:4 010030

0000 00	0	1	0	0	0	0	0 0 0 0 0011 0000

n=0, i=1, x=0, b=0, P=0, e=0

- Directed
- Immediate addressing mode.
- TA= Value of location

  = 030

A:5 003600

0000 00	0	0	0	0	1	1	0110 0000 0000

n=0, i=0, x=0, b=0, P=1, e=1

- Directed addressing modes
- PC relative.

TA = (PC) + displacement

= 003000 + 600 = 003600

A:6 0310C303

0000 0011 0001 0000

Instruction Set of SIC/XE — not only accumulator

Instructⁿ Set floating point, Base register

LDA, STA, LDB, STB

ADD, SUB, MUL, DIV, ADDF, SUBF, MULF, DIVF
ADDR, SUBR, MULR, DIVR. Register opera

COMP

JLT,   SVC Supervisor

JGT

I/O channels.

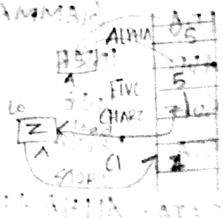
SIO - Start I/O channel

TIO - Test I/O channel

HIO - ~~Hold~~ Halt I/O channel.

SIC programming

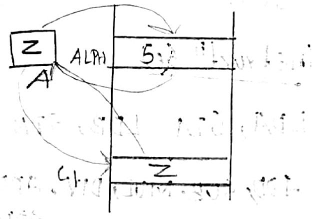
```
LDA   FIVE
STA   ALPHA
LDCH  CHARZ
STCH  C1        Reserve word
ALPHA  RESW 1  | CHARZ  BYTE  C'Z'   declarat
FIVE   WORD 5  | C1     RESB  1
```



Declaeakion is ak the end of the progmhg.

SIC/XE programming

```
LDA    #5
STA    ALPHA
LDCH   #90
STCH   CI
ALPHA  RESW 1
CI  RESB  1
```



Arithmetic Operations (SIC)

```
LDA   ALPHA        1 is added
ADD   INCR         ONE  WORD 1
SUB   ONE          ALPHA  RESW 1      1 is sub & loaded to AL
STA   BETA         BETA   RESW 1
LDA   GAMMA        GAMMA  RESW 1     Declaration
ADD   INCR         INCR   RESW 1
SUB   ONE          DELTA  RESW 1
STA   DELTA
```

o/p = BETA = ALPHA + INCR - 1

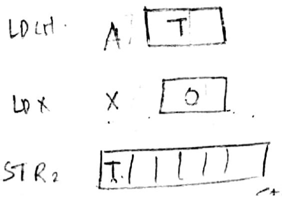
DELTA = GAMMA + INCR~~number~~ - word
character - - Byte

Arithmetic Operation (SIC/XE)

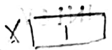
```
LDS   INCR                     ALPHA  RESW 1
LDA   ALPHA                    BETA   RESW 1
ADDR  S,A                      GAMMA  RESW 1
SUB   #1                       INCR   RESW 1
STA   BETA
LDA   GAMMA
ADDR  S,A
SUB   #1
STA   DELTA
```

Looping & Indexing (SIC)

```
        LDX   ZERO
MOVECH  LDCH  STR1,X          STR1    BYTE  C'TEST STRING'
        STCH  STR2,X          STR2    RESB  11
Increment
        TIX   ELEVEN          ZERO    WORD  0
        JLT   MOVECH          ELEVEN  WORD  11
```



TIX ELEVEN - Increment X



Then compare X and ELEVEN
if less than Jump. that is
again to MOVECH

2nd iteration

A $\boxed{E}$

X $\boxed{1}$

STR 2 $\boxed{T}\boxed{E}\boxed{\ }\boxed{\ }\boxed{\ }$

X $\boxed{2}$

SIC/XE , no need to declare constants.

```
          ~~LDX   #0~~
          LDT    #11
          LDX    #0
MOVECH    LDCH   STR1,X
          ~~LD~~ STCH  STR2,X
          TIXR   T
          JLT    MOVECH
STR1      BYTE   C'TEST STRING'
STR2      RESB   11
```

EQ => Ready

①
```
INLOOP  TD    INDEV
        JEQ   INLOOP    Ready
        RD    INDEV     Read, to Store
        STCH  DATA      Store the value.
OUTLP   TD    OUTDEV
        JEQ   OUTLP
        LDCH  DATA      Load
        WD    OUTDEV.   Write

INDEV   BYTE  X'F1'
OUTDEV  BYTE  X'05'
DATA    RESB  1
```

②
```
        LDA    ZERO       Load zero to accumulator
        STA    INDEX      Store zero.
label                     Store zero to index register.
ADDLP   LDX    INDEX
        LDA    ALPHA,X    Load alpha to accumulator
        ADD    BETA,X     Add beta + first letter of alpha
        STA    GAMMA,X    Result of addition
        LDA    INDEX
        ADD    THREE      3 byte = 1word to store
        STA    INDEX
        COMP   K300
        JLT    ADDLP
INDEX   RESW   1
ALPHA   RESW   100
BETA    RESW   100
GAMMA   RESW   100

ZERO    WORD   0
K30     WORD   300
```