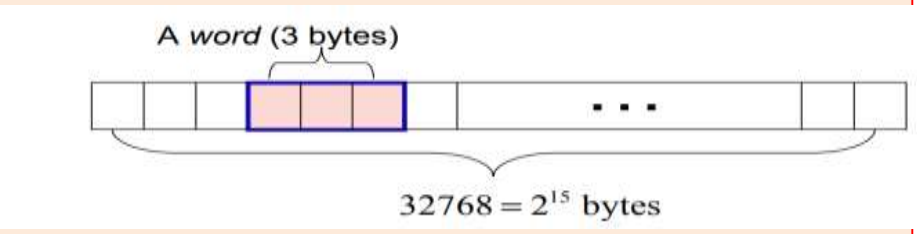
The Simplified Instructional Computer (SIC)

- SIC comes in two versions
  - The standard model
  - An XE version - “extra equipments” , “extra expensive”
- These two versions has been designed to be upward compatible
- An object program for the standard SIC will also execute properly on a SIC/XE system

SIC Machine Architecture

- **Memory**
  - 1 byte = 8-bit
  - 1 word=3 consecutive bytes
  - **Addressed by the location of their lowest numbered byte**
  - Total 32,768 (2^15) bytes
  - Memory is byte addressable



SIC does not have any stack.
It uses the linkage register to store the return address.
It is difficult to write the recursive program.
A programmer has to maintain memory for return addresses when we write more than one layer of function call.

Registers- Five registers (24 bits in length

Mnemonic	Number	Special use
A	0	**Accumulator** - used for arithmetic operations
X	1	**Index Register-** used for addressing
L	2	**Linkage register-** the Jump to Subroutine (JSUB) instruction stores the return address in this register
PC	8	**Program Counter** - contains the address of the next instruction to be fetched for execution
SW	9	**Status Word** - contains a variety of information, including a Condition Code (CC)

SIC Machine Architecture

- **Data Formats**
  - **Integers:** stored as 24-bit binary numbers;
    2's complement representation is used for negative values
  - **Characters:** stored as 8-bit ASCII codes
  - No floating-point hardware
- **Instruction Formats** - standard version of SIC

N ⇔ 2^n – N
Eg : if n = 4, -1 ⇔ 2^4 – 1 = (1111)2.

The flag bit x is used to indicate *indexed-addressing* mode

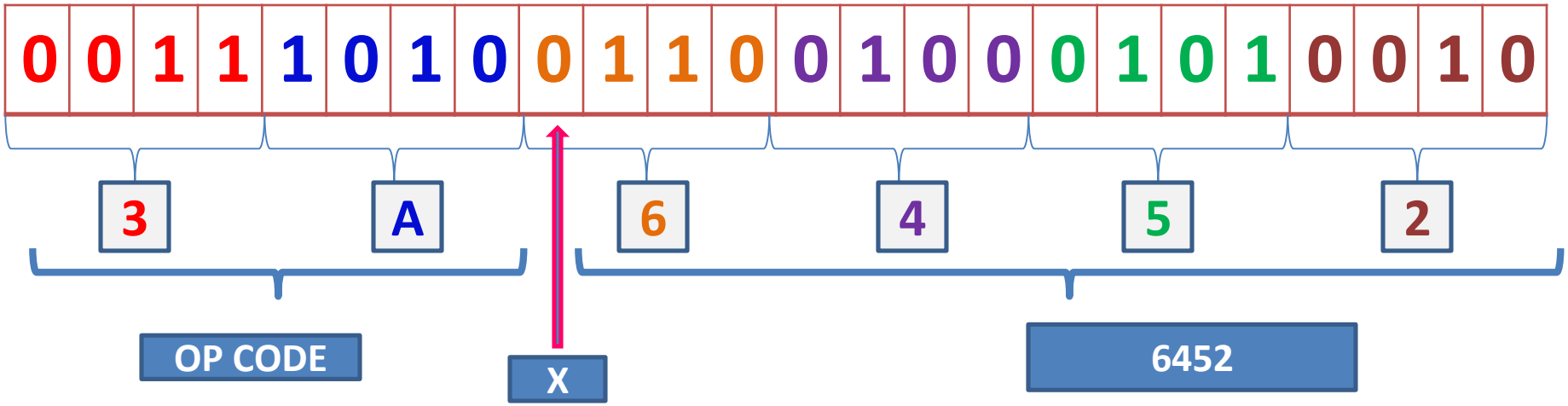
- **Addressing Modes**
  - There are two addressing modes available
  - Indicated by the setting of x bit in the instruction

Mode	Indication	Target address calculation
Direct	x=0	TA=address
Indexed	x=1	TA=address+(X)

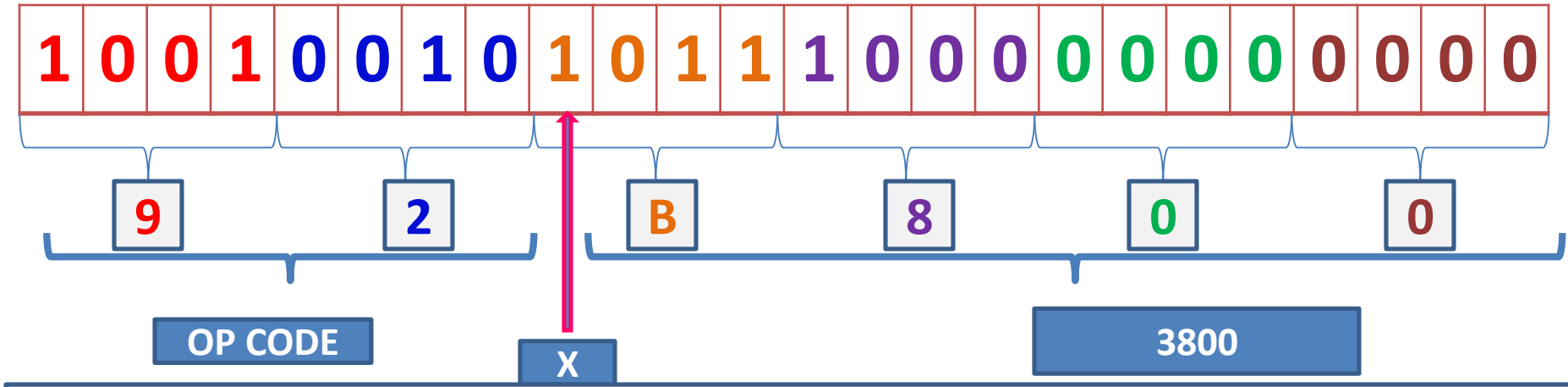
( ): the contents of a register or a memory location

Find EA corresponding to following instructions. Assume (X)= 2500

1) 3A6452 2) 92B800 3) 777777



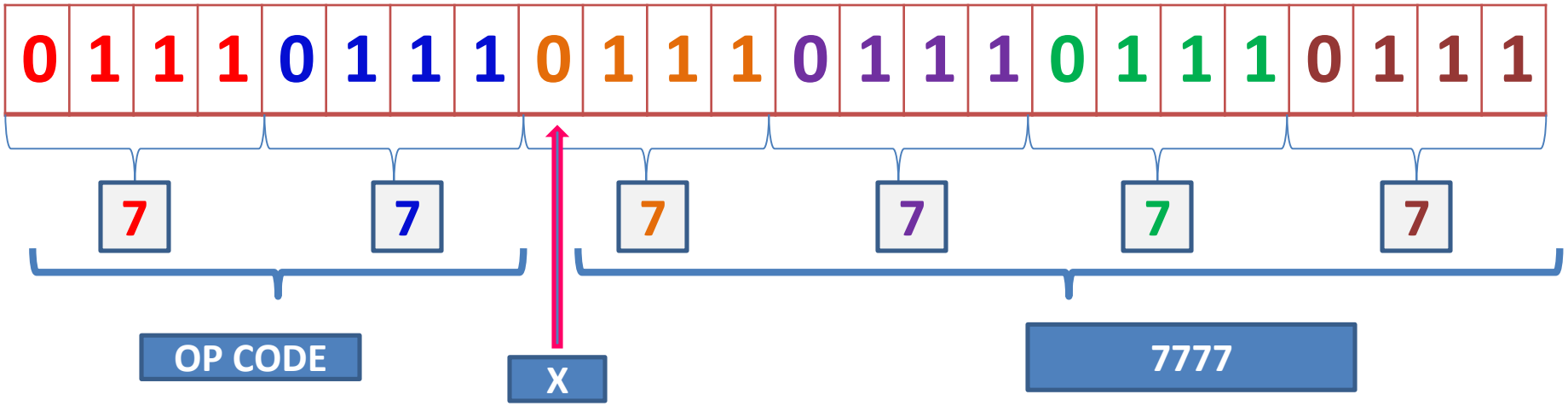
EA=15 bit address = 6452



EA=15 bit address +(X)= 3800+2500=5D00

Find EA corresponding to following instructions. Assume (X)= 2500

1) 3A6452 2) 92B800 3) 777777



EA=15 bit address+ (X)= 6452+7777=DBC9

Instruction set

Classified into 7 Categories

1. Load and Store Instructions
2. Arithmetic Instruction
3. Logical Instruction
4. Compare Instruction
5. Jump Instruction
6. Subroutine handling Instruction
7. I/O instruction

ADD
SUB
MUL
DIV

Involve **register A** and **a word in memory**

COMP

involves **register A** and **a word in memory** save **result in condition code (CC)** of SW

AND
OR

J
JEQ
JGT
JLT

LDA
LDCH
LDL
LDX
STA
STCH
STL
STSW
STX

JSUB
RSUB

TD
WD
RD

tests whether the **addressed device is ready** to send or receive a byte of data

CC : < : ready

CC : = : busy

Load and Store Instructions

MNEMONIC	OPERAND	OPCODE	EXPLANATION
LDA	M	00	**A = M**
LDCH	M	50	**A[RMB] = [RMB]**
LDL	M	08	**L = M**
LDX	M	04	**X = M**
STA	M	0C	**M = A**
STCH	M	54	**M[RMB] = A[RMB]**
STL	M	14	**M = L**
STSW	M	E8	**M = SW**
STX	M	10	**M = X**

Notations used

A - Accumulator M - Memory CC - Condition Code

PC - Program Counter RMB - Right Most Byte L - Linkage Register

Arithmetic and Logical Instruction

MNEMONIC	OPERAND	OPCODE	EXPLANATION
ADD	M	18	**A = A + M**
SUB	M	1C	**A = A – M**
MUL	M	20	**A = A * M**
DIV	M	24	**A = A / M**
Logical Instruction			
AND	M	40	**A = A AND M**
OR	M	44	**A = A OR M**
Compare Instruction			
COMP	M	28	**compares A and M**
TIX	**M**	**2C**	**X = X + 1 ; compare X with M**

Looping (TIX)

- (X)=(X)+1
- compare with operand
- set CC

Notations used

A - Accumulator M - Memory CC - Condition Code

PC - Program Counter RMB - Right Most Byte L - Linkage Register

Jump Instructions

MNEMONIC	OPERAND	OPCODE	EXPLANATION
J	M	3C	**PC = M**
JEQ	M	30	**if CC set to =, PC = M**
JGT	M	34	**if CC set to >, PC = M**
JLT	M	38	**if CC set to <, PC = M**
Subroutine handling Instructions			
JSUB	M	48	**L = PC ; PC = M**
RSUB		**4C**	**PC = L**

Notations used

A - Accumulator **M - Memory** **CC - Condition Code**

PC - Program Counter **RMB - Right Most Byte** **L - Linkage Register**

I/O instructions

MNEMONIC	OPERAND	OPCODE	EXPLANATION
TD	M	E0	**test device specified by M**
WD	M	DC	**device specified by M[RMB] = S[RMB]**
RD	M	D8	**A[RMB] = data specified by M[RMB]**

I/O operation is performed by transferring 1 byte at a time from or to rightmost 8 bits of accumulator.

Each device has 8 bit unique code (Device Address)

There are 3 I/O instructions:

Test Device (TD) tests whether device is ready or not. Condition code in Status Word Register is used for this purpose. If **cc is <** then **device is ready** otherwise device is busy.

Read data(RD) reads a byte from device and stores in register A.

Write data(WD) writes a byte from register A to the device.

Notations used

A - Accumulator M - Memory CC - Condition Code

PC - Program Counter RMB - Right Most Byte L - Linkage Register

Assembler Directives

Pseudo-Instructions

- Not translated into machine instructions
- Providing information to the assembler

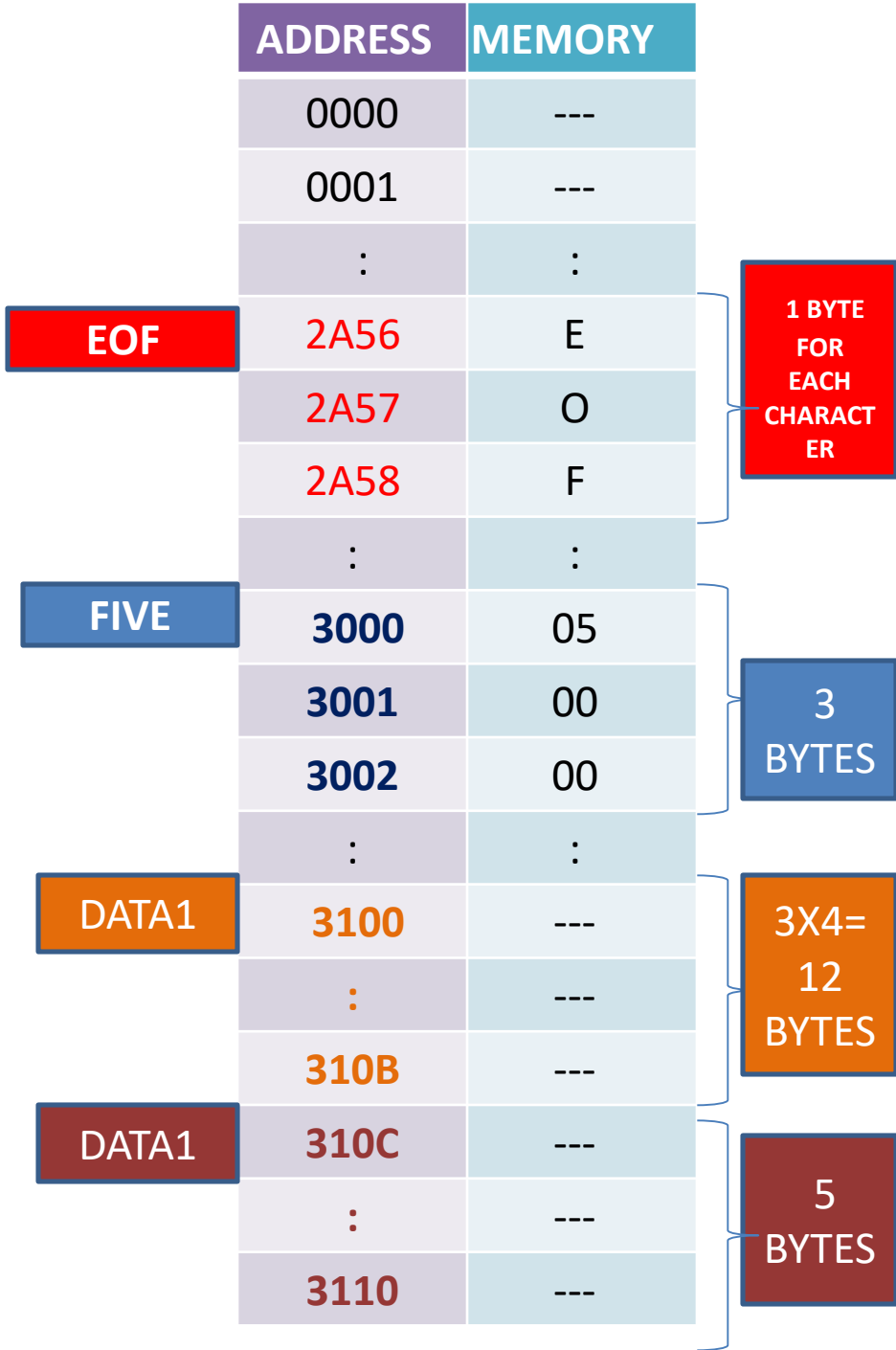
START	Specify name and starting address for the program.
END	Indicate the end of the source program and (optionally) specify the first executable instruction in the program.
BYTE	Generate character or hexadecimal constant, occupying as many bytes as needed to represent the constant.
WORD	Generate one-word integer constant.
RESB	Reserve the indicated number of bytes for a data area.
RESW	Reserve the indicated number of words for a data area.

Syntax

Label START value
Label BYTE value
Label WORD value
Label RESB value
Label RESW value

Label: name of operand
value: integer, character

Eg. EOF BYTE C'EOF'
B1 BYTE X'4156'
FIVE WORD 5
DATA1 RESW 4
DATA2 RESB 5



Assume that to memory location named FIVE and CHARX contains data 5 and 'Z' respectively. Write sequence of statement to transfer content of location FIVE and CHARZ to location ALPHA and C1 respectively

```
          START   1000
1000      LDA     FIVE      load 5 into A
1003      STA     ALPHA     store in ALPHA
1006      LDCH    CHARZ     load 'Z' into A
1009      STCH    C1        store in C1
100C      RSUB
```

```
ALPHA   100F  RESW   1       reserve one word space
FIVE    1012  WORD   5       one word holding  5
CHARZ   1015  BYTE   C'Z'    one-byte constant
C1      1016  RESB   1       one-byte variable
        1017  END
```

Arithmetic operations: BETA = ALPH +INCR-1

```
0000        PG1     START
0000                LDA     ALPH
0003                ADD     INCR
0006                SUB     ONE
0009                STA     BETA
000C                RSUB
000F        ONE     WORD    1       one-word constant
0012        ALPH    RESW    1       one-word variables
0015        BETA    RESW    1
0018        INCR    RESW    1
001B                END
```

To copy a 11 byte string from one location to another

```
        LDX     ZERO            Initializes X to zero.
MOVECH  LDCH    STR1,X          X  specifies indexing.
        STCH    STR2,X
        TIX     ELEVEN          Increments X and compares with 11.
        JLT     MOVECH
        RSUB
STR1    BYTE    C'TEST STRING'  String constant.
STR2    RESB    11
ELEVEN  WORD    11
ZERO    WORD    0
```

- SUBROUTINE TO READ 100-BYTE RCORD

	START	1000	
READ	LDX	ZERO	INITAILIZE INDEX REGISTER TO 0
RLOOP	TD	INDEV	TEST INPUT DEVICE
	JEQ	RLOOP	LOOP IF DEVICE IS BUSY
	RD	INDEV	READ ONE BYTE INTO REGISTER A
	STCH	RECORD,X	STORE DATA BYTE INTO RECORD
	TIX	K100	**ADD 1 TO INDEX AND COMPARE TO 100**
	JLT	RLOOP	LOOP IF INDEX IS LESS THAN 100
	RSUB		EXIT FROM SUBROUTINE
ZERO	WORD	0	
K100	WORD	100	INPUT DEVICE NUMBER
INDEV	BYTE	X'F1'	100-BYTE BUFFER FOR INPUT RECORD
RECORD	RESB	100	ONE-WORD CONSTANTS
	END		

	LDA	ZERO
	STA	INDEX
LOOP1	LDA	INDEX
	LDA	ALPHA,X
	ADD	BETA,X
	STA	GAMMA,X
	LDA	INDEX
	ADD	THREE
	STA	INDEX
	COMP	D300
	JLT	LOOP1
	: :	
INDEX	RESW	1
ZERO	WORD	0
D300	WORD	300
THREE	WORD	3

GAMMA(I)=ALPHA[I]+BETA[I]

ALPHA	RESW	100
BETA	RESW	100
GAMMA	RESW	100
	END	

SIC/XE

Memory

- Almost the same as that previously described for SIC
- However, 1 MB $(2^{20}$ bytes) maximum memory available

Byte organized WORD

Registers

- More registers are provided by SIC/XE

Mnemonic	Number	Special use
B	3	Base register
S	4	General working register
T	5	General working register
F	6	Floating-point accumulator (48 bits)

Registers common to SIC and SIC/XE

A	0
X	1
L	2
PC	8
SW	9

Data Formats

The same data format as the standard version

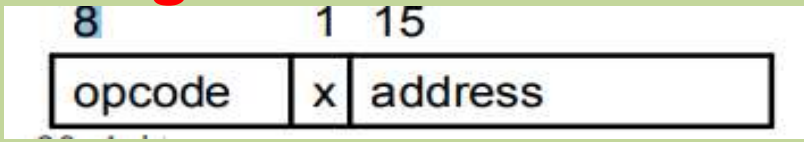
However, provide an addition 48-bit floating-point data type

- **fraction: between 0 and 1**
- **exponent: Value between 0 to2047**
- **sign: 0=positive, 1=negative**

$$\text{Value} = (-1)^S\ 0.f * 2^{(exp-1024)}$$

- **Instruction formats**

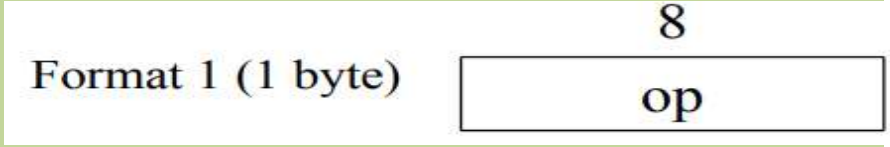
- Since the memory used by SIC/XE may be **2^20** bytes, the instruction **format of SIC is not enough**.

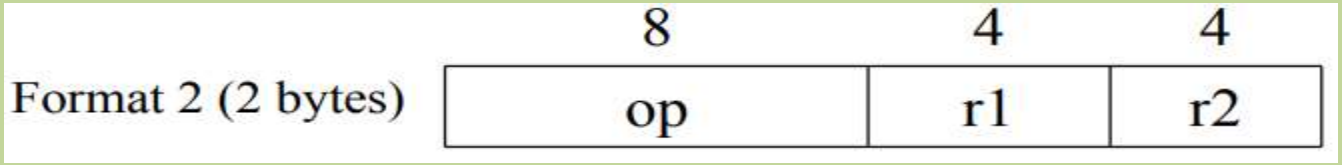


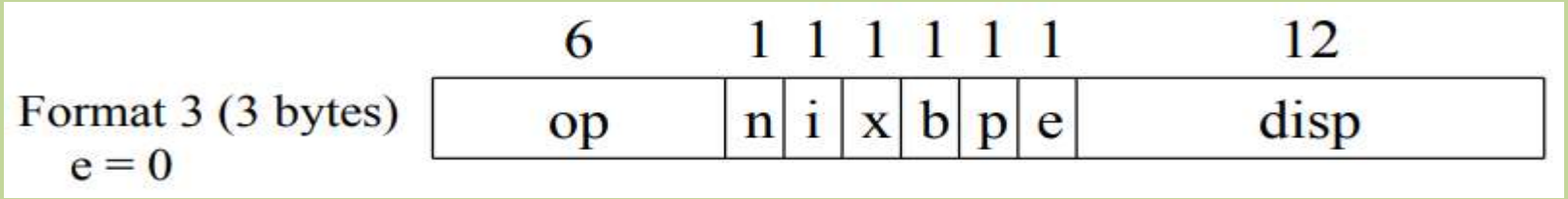
Solutions

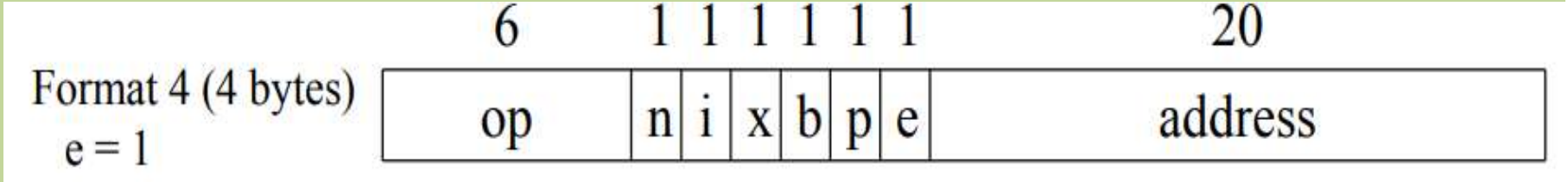
- Use relative addressing
- Extend the address field to 20 bits

SIC/XE instruction formats









Addressing Modes

Base relative addressing **- format 3 only**

- n = 1, i = 1, ***b=1, p=0***

Program-counter relative addressing **- format 3 only**

- n = 1, i = 1, ***b=0, p=1***

Direct addressing **– format 3 and 4**

- n = 1, i = 1, ***b=0, p=0***

Indexed addressing **– format 3 and 4**

- n = 1, i = 1, ***x = 1*** or n = 0, i = 0, ***x = 1***

Immediate addressing **– format 3 and 4**

- **n = 0, i = 1,** x = 0 // cannot combine with *indexed*

Indirect addressing **– format 3 and 4**

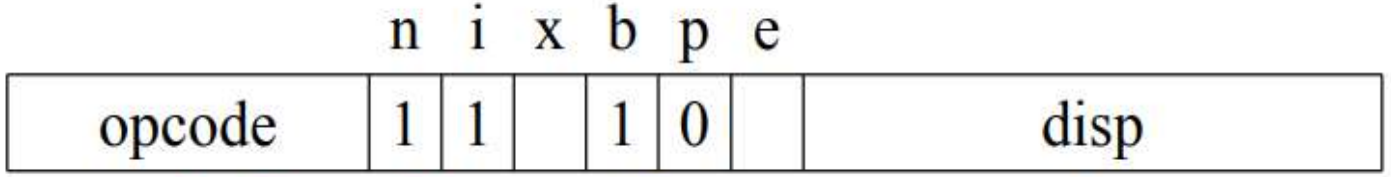
- **n = 1, i = 0,** x = 0 // cannot combine with *indexed*

Simple addressing **– format 3 and 4**

- **n = 0, i = 0 or n = 1, i = 1**

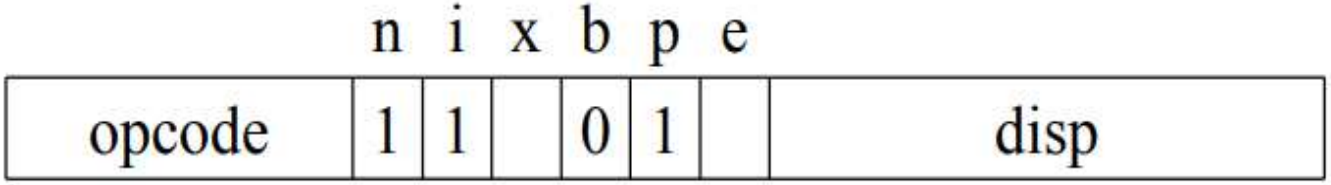
Addressing Modes: Address Computation

- ***Base Relative Addressing***



n=1, i=1, ***b=1, p=0***, TA=(**B**)+disp (0≤disp ≤4095)

- ***Program-Counter Relative Addressing***



n=1, i=1, ***b=0, p=1***, TA=(**PC**)+disp (-2048≤disp ≤2047)

Addressing Modes: Address Computation

- ***Direct Addressing***
  - The target address is taken directly from the *disp* or *address* field

	n	i	x	b	p	e	
opcode	1	1		0	0		disp/address

Format 3 (e=0): n=1, i=1, ***b=0, p=0***, TA=disp (0≤disp ≤4095)

Format 4 (e=1): n=1, i=1, ***b=0, p=0***, TA=address

- ***Indexed Addressing***
  - The term (X) is added into the target address calculation

	n	i	x	b	p	e	
opcode	1	1	1				disp/address

n=1, i=1, ***x=1***

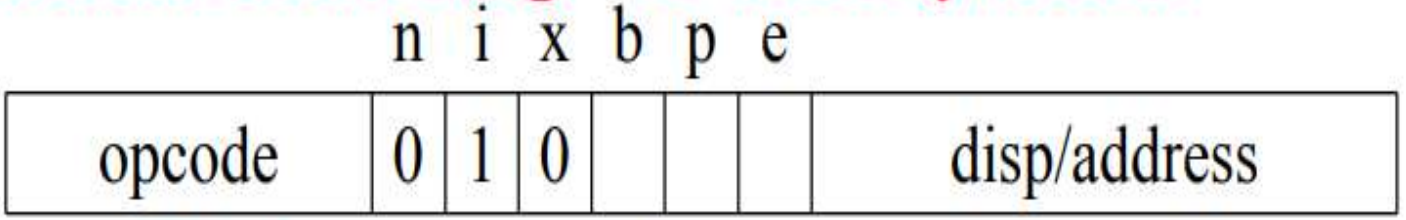
Ex. *Direct Indexed Addressing*

Format 3, TA=(X)+disp

Format 4, TA=(X)+address

Addressing Modes: Address Computation

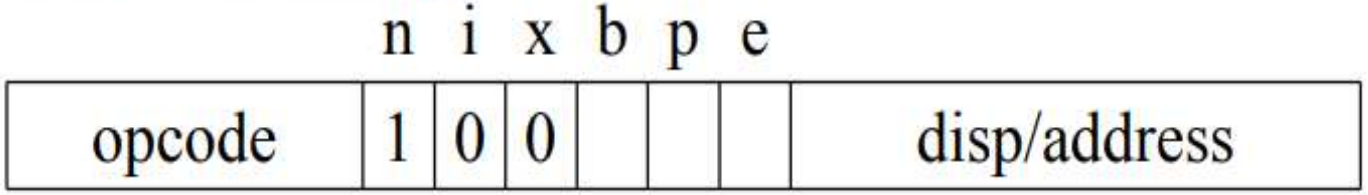
- ***Immediate Addressing – no memory access***



n=0, i=1, x=0, **operand**=disp //format 3

n=0, i=1, x=0, **operand**=address //format 4

- ***Indirect Addressing***

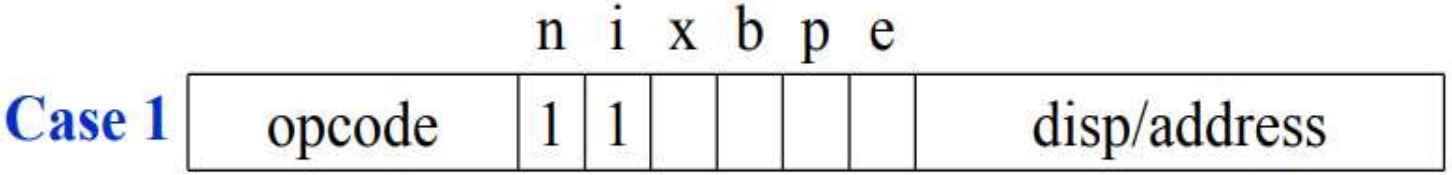


n=1, i=0, x=0, TA=(disp), **operand** = (TA) = ((disp))

n=1, i=0, x=0, TA=(address), **operand** = (TA) = ((address))

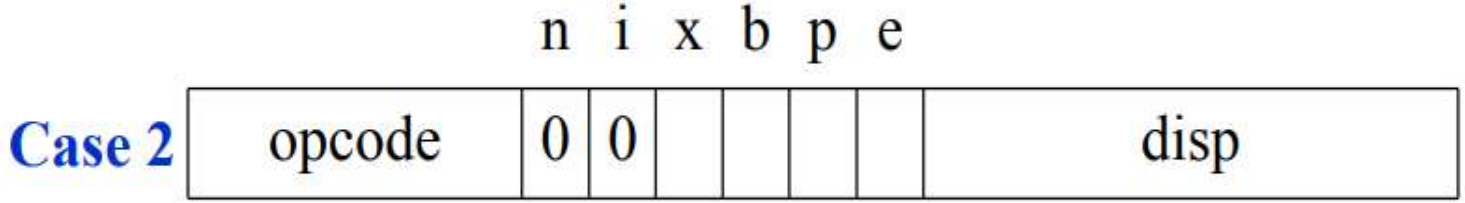
Addressing Modes: Address Computation

- ***Simple Addressing Mode***



Format 3: ***i=1, n=1***, TA=disp, **operand** = (disp)

Format 4: ***i=1, n=1***, TA=address, **operand** = (address)

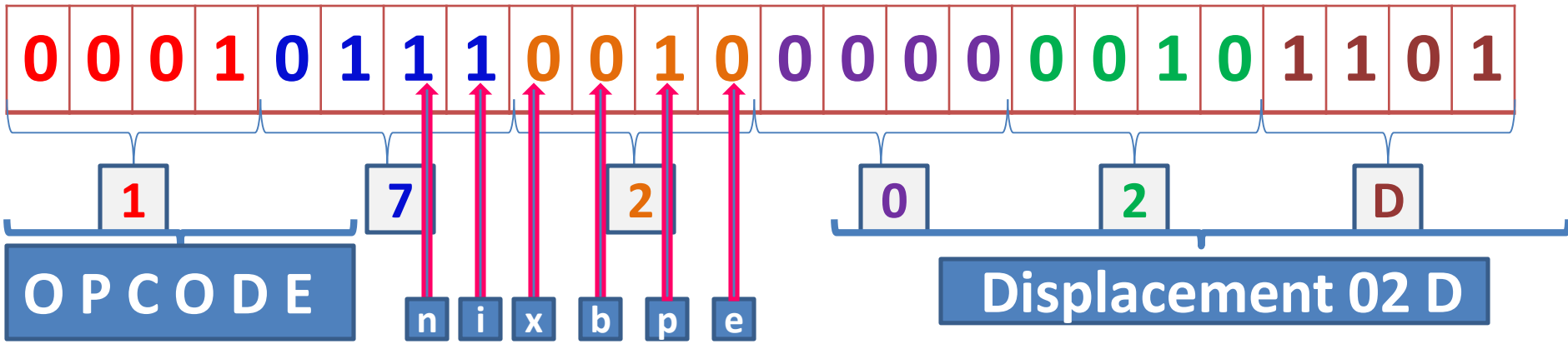


i=0, n=0, TA=b/p/e/disp (SIC standard)

- Equal with a special hardware feature to provide the upward compatibility
  - If bits $n = i = 0$,
    - Neither immediate nor indirect
    - A special case of ***Simple Addressing***
    - Bits *b*, *p*, *e* are considered to be ***part*** *of the address field* of the instruction
    - Make Instruction Format 3 identical to the format used in SIC
    - Thus, provide the desired compatibility

Find EA corresponding to following instructions. Assume (X)= 2500

1) 17202D 2) 92B800 3) 777777



Assume (PC) = 3

OPCODE=00010100=14

Format = 3

n i = 1 1 (DIRECT ADDRESS)

P=1

EA= (PC)+ Displacement

=003+02D=0030

http://cis.csuohio.edu/~jackie/cis335/sicxe_address.txt