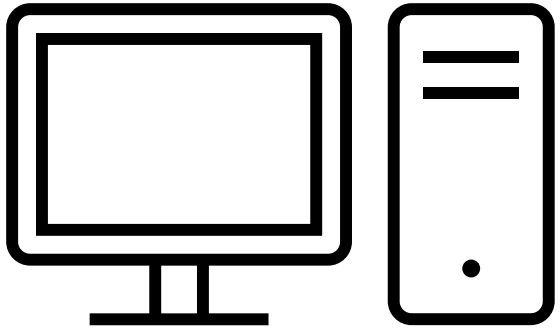
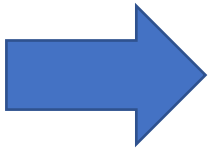
시스템 소프트웨어
SIC & SIC/XE

SIC와 SIC/XE를 사용하는 이유?



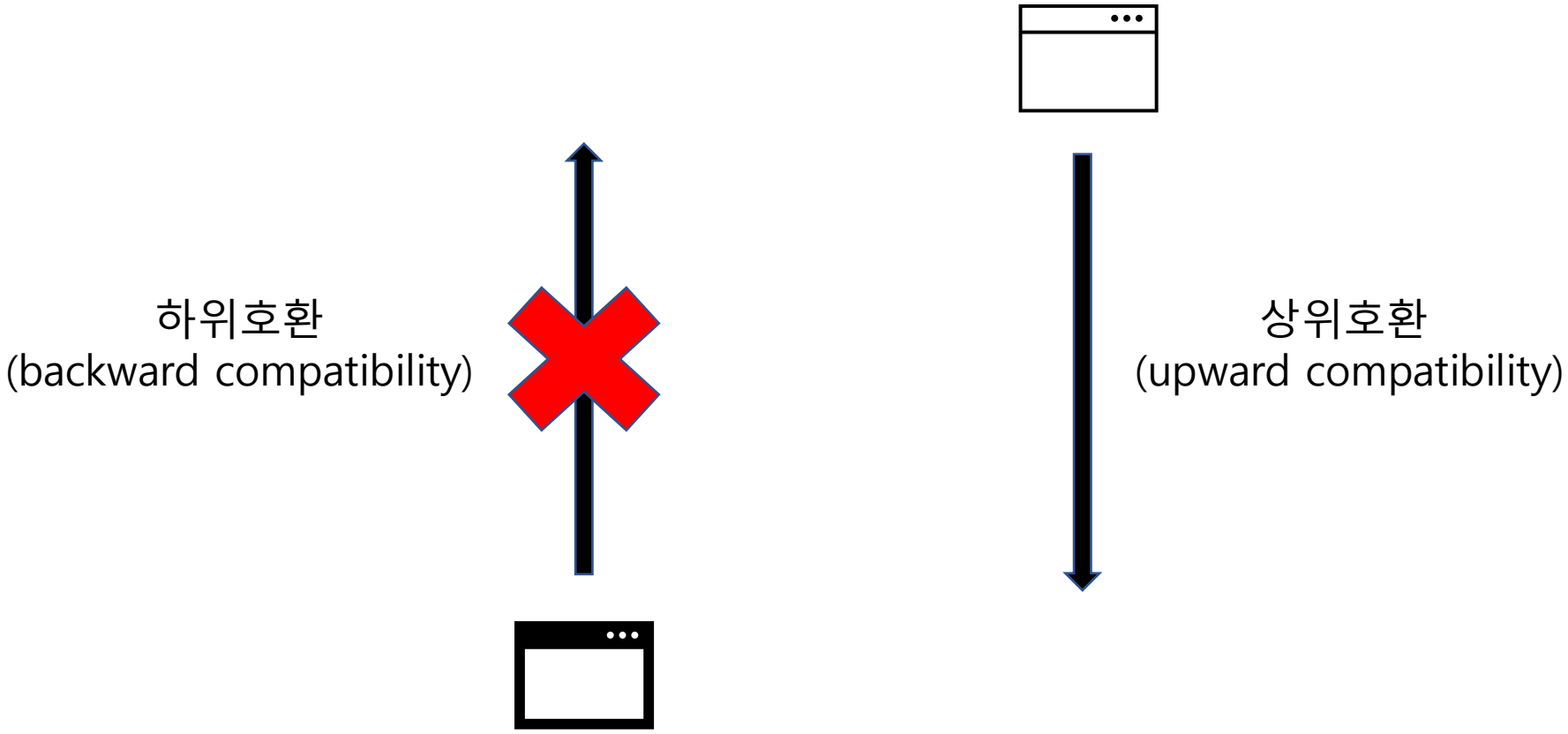
실제 머신



```
COPY    START   1000            COPY FILE FROM INPUT TO OUTPU
FIRST   STL     RETADR          SAVE RETURN ADDRESS
CLOOP   JSUB    RDREC           READ INPUT RECORD
        LDA     LENGTH          TEST FOR EOF (LENGTH = 0)
        COMP    ZERO
        JEQ     ENDFIL          EXIT IF EOF FOUND
        JSUB    WRREC           WRITE OUTPUT RECORD
        J       CLOOP           LOOP
ENDFIL  LDA     EOF             INSERT END OF FILE MARKER
        STA     BUFFER
        LDA     THREE           SET LENGTH = 3
        STA     LENGTH
        JSUB    WRREC           WRITE EOF
        LDL     RETADR          GET RETURN ADDRESS
        RSUB                    RETURN TO CALLER
EOF     BYTE    C'EOF'
THREE   WORD    3
ZERO    WORD    0
RETADR  RESW    1
LENGTH  RESW    1               LENGTH OF RECORD
BUFFER  RESB    4096            4096-BYTE BUFFER AREA
```

SIC, SIC/XE

SIC - **S**implified **I**nstructional **C**omputer

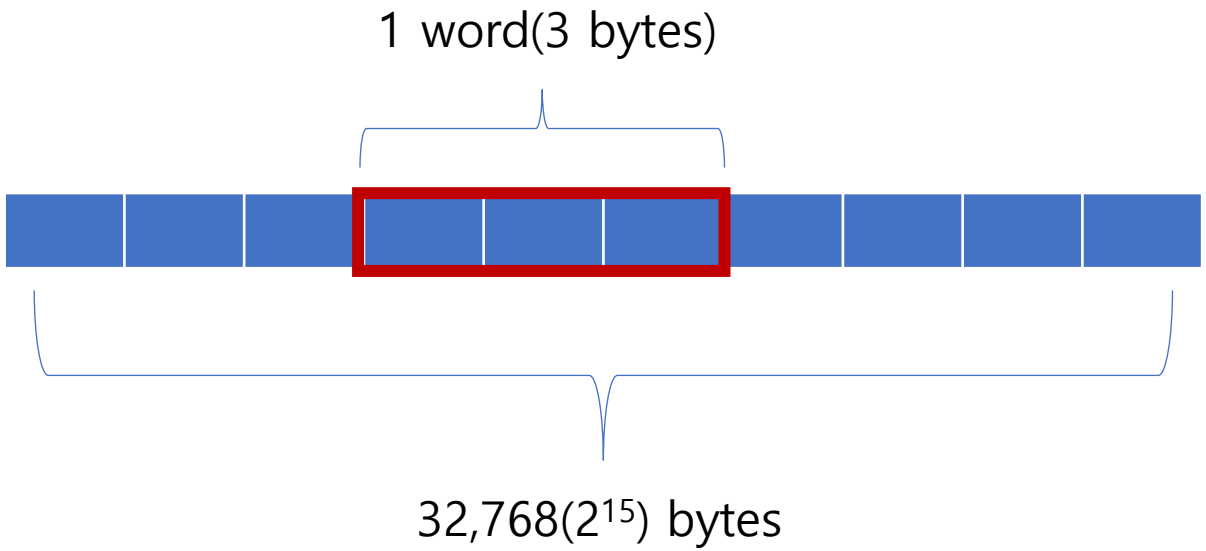


SIC/XE - **S**implified **I**nstructional **C**omputer**/**the e**X**tra **E**quipment

SIC - Simplified Instructional Computer

< Memory >

- 1 byte = 8bits
- 모든 memory address는 byte addresses로 표현되며 저장된다.
- 3 bytes = 1 word
- SIC의 전체 memory = 32,768(2^{15}) bytes



SIC - Simplified Instructional Computer

< Registers >

1. A(0) : Accumulator register
→ 기본적인 산술 연산에 활용

2. X(1) : Index register
→ 주소를 저장하고 계산하는데 사용

3. L(2) : Linkage register
→ Jump to Subroutine 시, return address를 저장

4. PC(8) : Program Counter register
→ 다음에 실행할 명령어의 주소를 저장

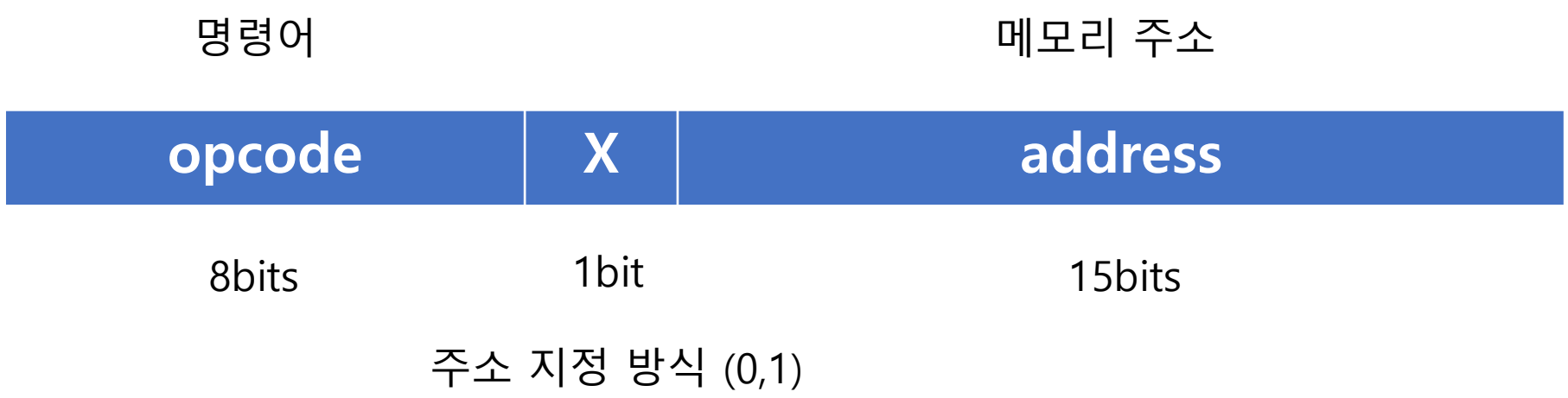
5. SW(9) : status Word register
→ 산술 연산 결과의 상태를 알려주는 플래그 비트들을 저장

SIC - Simplified Instructional Computer

< Data format >

1. Integer : 24 bits binary numbers
2. Character : 8 bits ASCII code

< Instruction format >



SIC - Simplified Instructional Computer

< 주소 지정 방식 >

X = 0 → 직접 주소 지정 방식

: Target Address = address

X = 1 → 인덱스 주소 지정 방식

: Target Address = address + (X) * (X)는 X 레지스터 안의 값을 나타냄

SIC - Simplified Instructional Computer

< Instruction Type >

표기법
A ← (A) + (m) : A 레지스터에 A 레지스터 값과 메모리 주소 m의 값을 저장한다.

1. Load : memory → register

   Store : register → memory

LDA m : A register ← (m) * (m) = value of m

LDX m : X register ← (m)

STA m : m ← (A register) *(A register) = value of A register

STX m : m ← (X register) *(X register) = value of X register

SIC - Simplified Instructional Computer

< Instruction Type >

2. Arithmetic & Logic

ADD m : A ← (A) + (m)

SUB m : A ← (A) - (m)

MUL m : A ← (A) * (m)

DIV m : A ← (A) / (m)

AND m : A ← (A) & (m)

OR m : A ← (A) | (m)

SIC - Simplified Instructional Computer

< Instruction Type >

3. Comparison

COMP m : A register의 값과 m의 값을 비교하여 나온 값(크다, 작다, 같다)을 SW register에 저장

TIX m : X register의 값에 1을 더한 후, m의 값과 비교하여 나온 값(크다, 작다, 같다)을 SW register에 저장

4. Conditional Jumps

J m : SW register의 값과 관계없이 PC ← m

JLT m : COMP 명령어에서 '작다'가 SW register에 저장되었다면 PC ← m if CC set to <

JGT m : COMP 명령어에서 '크다'가 SW register에 저장되었다면 PC ← m if CC set to >

JEQ m : COMP 명령어에서 '같다'가 SW register에 저장되었다면 PC ← m if CC set to =

SIC - Simplified Instructional Computer

< Instruction Type >

5. Subroutine Linkage

JSUB m : L register에 PC에 저장되었던 명령어를 저장하고, PC에 m이란 subroutine을 저장한다.
L ← (PC); PC ← m

RSUB : subroutine(m)이 끝난 후, 원래 다음 명령어를 L register를 참조해 돌아온다.
PC ← (L)

6. I/O

TD : I/O Device를 사용하기 전에 사용 가능한지 확인하는 명령어

RD : I/O Device에 1 Byte를 읽는 명령어

WD : I/O Device에 1 Byte를 쓰는 명령어

SIC - Simplified Instructional Computer

< Instruction Type >

7. Define Storage

CHARA BYTE C'A' : CHARA는 A를 값으로 하는 상수 선언

FOUR WORD 4 : FOUR는 4를 값으로 하는 상수 선언

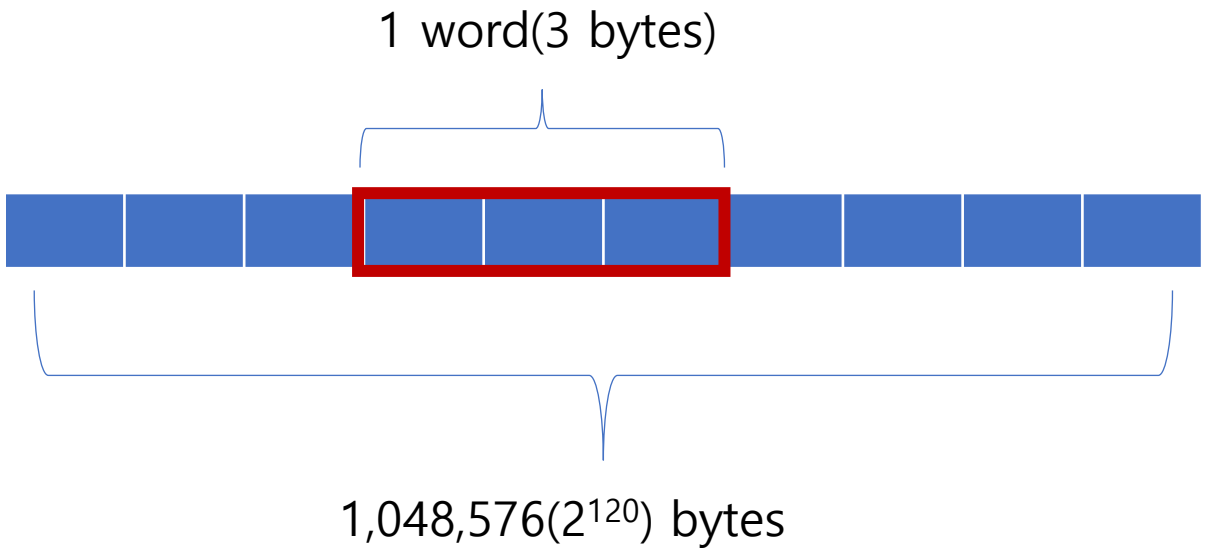
TEMP RESB 2 : 2 BYTE 만큼 TEMP에 예약

TEMP RESW 1 : 1 WORD 만큼 TEMP에 예약

SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Memory >

- 1 byte = 8bits
- 모든 memory address는 byte addresses로 표현되며 저장된다.
- 3 bytes = 1 word
- SIC/XE의 전체 memory = 1MB(2^{20}) bytes



SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Registers >

1. A(0), X(1), L(2), PC(8), SW(9)

2. B(3) : Base register
→ Addressing시 사용

3. S(4) : General working register
→ 특별한 용도 없는 범용 register로 사용

4. T(5) : General working register
→ 특별한 용도 없는 범용 register로 사용

5. F(6) : Floating Point Accumulator
→ 실수 연산을 하기 위한 register

SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Data format >

1. Integer, Character
2. Floating Point : 48 bits

$$(-1)^{s} * \text{Fraction} * 2^{(\text{Exponent}-1024)}$$

S	Exponent	Fraction
1bit	11bits	36bits
S = 0 : 양수 S = 1 : 음수	0 ≤ Exponent ≤ 2047	0 ≤ Fraction ≤ 1

SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Instruction format >

format 1
: opcode만 사용하는 명령어 (메모리에 reference 하지 않음.)
ex) FIX : F register의 값을 정수로 변환해 A register에 저장.

format 2
: opcode와 2개의 register 주소를 사용하는 명령어
Ex) ADDR T A : T register의 값과 A register의 값을 더해서 A register에 저장.

SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Instruction format >

Format 3
: opcode와 6개의 flag, disp를 사용하는 명령어

Mode	Indication	Target address calculation
Base relative mode	b = 1, p = 0	TA = B register + disp (0 ~ 4095)
PC relative mode	b = 0, p = 1	TA = PC register + disp (-2048 ~ 2047)
Direct Addressing mode	b = 0, p = 0	TA = disp

format 4
: opcode, 6개의 flag, address를 사용하는 명령어

Mode	Indication	Target address calculation
Immediate Addressing mode	n = 0, i = 1	TA의 값을 주소 값으로 사용하지 않고, 그 자체를 값으로 사용.
Indirect Addressing mode	n = 1, i = 0	TA를 간접 주소로 사용
Simple Addressing mode	n = 0, i = 0 n = 1, i = 1	TA로 메모리에 접근하여 값을 사용.

SIC/XE - Simplified Instructional Computer/the eXtra Equipment

< Instruction Type >

1. Load & Store
: register의 추가로 Load, Store 명령어들이 추가됨 ex) LDB, STB …

2. Floating Point Arithmetic
: 실수 연산이 가능해져, 실수의 사칙연산 명령어들이 추가됨 ex) ADDF, SUBF, MULF, DIVF …

3. Register instruction
: 범용 register의 추가로 register간의 연산이 가능해짐 ex) RMO, ADDR, SUBR, MULR, DIVR …

4. I/O
: SIC/XE로 되면서 I/O Device에서 값을 입력받거나 입력하면서 동시에 연산도 가능해짐. ex) SIO, TIO, HIO …

감사합니다.

Email : kdsvip5@naver.com