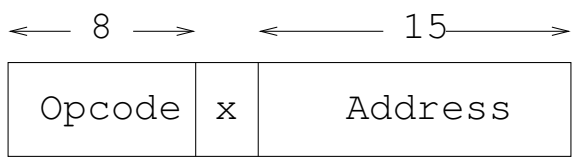
CSI 402 – Lecture 3
(SIC and SIC/XE Assembly Languages)

SIC and SIC/XE Architectures

Ref: Chapter 1 and Appendix A of [Beck].

Details regarding SIC:

- Allows only integers and characters.
- Memory size $= 2^{15}$ bytes. (Byte address: 15 bits).
- Word size $= 3$ bytes (or 24 bits).
- Characters: 1 byte (ASCII).
- Integers use 3 bytes; 2's complement for negative values.
- Five registers (denoted by A, X, L, PC and SW).
- **Instruction Format:**



Note: Here, `x` is the "index bit".

Details Regarding SIC (continued)

(a) Examples of Load/Store Instructions: Here, `m` denotes a memory address.

```
LDA    m
STA    m
STX    m
```

(b) Examples of Arithmetic Instructions: Here, register A is always an operand. Also, the result is in A.

```
ADD    m
DIV    m
```

(c) Comparison Instruction:

```
COMP    m
```

Note: Compares the contents of A with the contents of the specified memory location; sets CC to indicate $>$, $=$ or $<$.

Details Regarding SIC (continued)

(d) Conditional Jump Instructions:

```
JLT    m
JEQ    m
JGT    m
```

(e) Procedure call/return Instructions:

```
JSUB    m
RSUB
```

(f) I/O Instructions:

```
RD    d
WD    d
TD    d
```

Notes:

- Each I/O device has an 8-bit ID.
- I/O is done using register A one byte at a time.

Details Regarding SIC (continued)

Outline for I/O in SIC:

1. **repeat**
   Test device.
   until device is ready.
2. Read/write one byte.

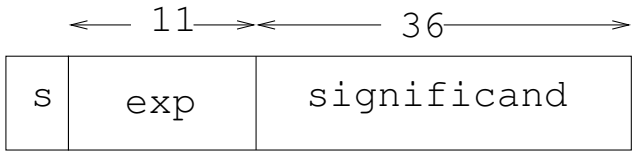
Note: This method of I/O is called "Busy-Wait" or "Programmed" I/O.

Details regarding SIC/XE:

- Allows integers, characters and floating point values.
- Memory size = 2^{20} bytes. (Byte address: 20 bits).
- Word size = 3 bytes (or 24 bits).
- Four additional registers (denoted by B, S, T and F).

Details Regarding SIC/XE (continued)

Floating point format:



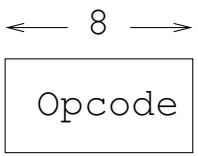
Note: The 11-bit exponent (exp) is stored in Biased-1024 form.

Instruction Formats for SIC/XE:

- Addresses must be 20 bits long.
- Uses both relative addressing and extended addressing.
- Instructions may be 1, 2, 3 or 4 bytes long.

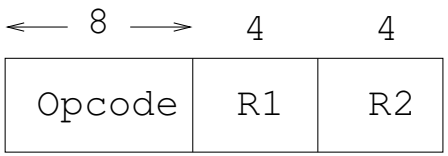
Instruction Formats for SIC/XE (continued)

1-Byte format:



Example: `FLOAT`

2-Byte format:

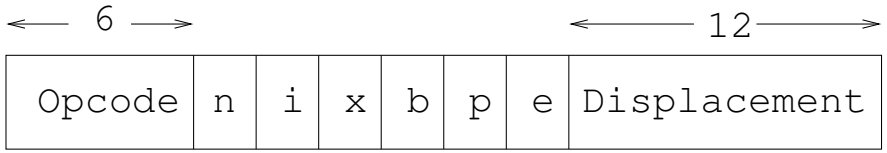


Examples:

```
RMO     S, T
MULR    S, T
```

Instruction Formats for SIC/XE (continued)

3-Byte format:



Notes:

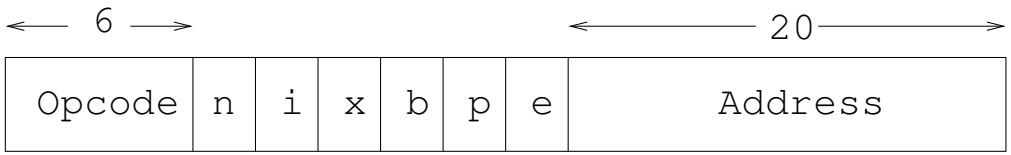
- x – index bit (as before).
- Bits b and p specify **relative** addressing.
  - b = 1 and p = 0: Relative to B.
  - b = 0 and p = 1: Relative to PC.
  - b = 0 and p = 0: Direct address.
- In B-relative mode, displacement is treated as an unsigned integer.
- In PC-relative mode, displacement is treated as an integer in 2's complement form.

Instruction Formats for SIC/XE (continued)

3-Byte Format (continued)

- Bits i and n specify extended addressing modes.
  - i = 1 and n = 0: **Immediate** mode.
  - i = 0 and n = 1: **Indirect** mode.
  - i = n: **Simple** addressing.
- Bit e is always 0 for 3-byte instructions.

4-Byte format:



Notes:

- Bits b and p are always 0. (Thus, 4-byte format does not allow relative addressing.)
- Bit e is always 1 (to distinguish between 3-byte and 4-byte instructions).

Instruction Formats for SIC/XE (continued)

SIC/XE Instruction Set:

- Includes all SIC instructions.
- New instructions (e.g. LDB, STB).
- Floating point instructions (e.g. ADDF, MULF).
- Register-to-register instructions (e.g. ADDR, MULR).

Examples of Address Calculation: To be presented in class.

Program Examples: Handout 3.1.

Suggested Exercise: Example 2 of Handout 3.1 shows the SIC version of an assembly language program. Write the SIC/XE version of the program.

Review of some Architectures

- Brief discussion on MIPS, Pentium and SPARC.
- Others from [Beck] left as reading assignment.

MIPS Architecture (Review):

- Used in SGI workstations.
- Load/store architecture; number of instructions is around 100 (RISC).
- Memory address = 32 bits; word size = 32 bits; processor can support both big endian and little endian ordering.
- Instruction length = 32 bits.
- Separate co-processors C0 (for exception handling) and C1 (floating point operations).
- 32 CPU registers for integer operations; each register is 32 bits long.

Review of MIPS (continued)

- 32-bit (single precision) and 64-bit (double precision) floating point registers are available; floating point representation based on IEEE Floating Point Standard (FPS).
- Simple addressing modes (immediate, base + displacement and PC-relative); load address (`la`) instruction to support indirect addressing.
- Supports all basic data types of C (`int`, `char`, `float` and `double`).

IA-32 Architecture (Intel):

- Started with Intel 80386 and has continued up to Itanium (64 bit CPU).
- Memory address = 32 bits; little endian; word size = 32 bits.
- 8 general purpose 32-bit registers (R0 through R7) and 8 floating point 64-bit registers (FP0 through FP7).

IA-32 Architecture (continued)

- Instruction Pointer (PC) and Status Register (containing condition code).
- More than 400 instructions; instruction lengths vary from 1 to 12 bytes (CISC).
- A variety of addressing modes (e.g. immediate, direct, register, register indirect, base + displacement, base + index, etc).
- Supports all basic data types of C. Also supports `quad` precision for floating point.
- Separate floating point processor.

Sun SPARC Architecture

- Used in Sun workstations. (Designed in 1987 and has evolved to SuperSPARC, UltraSPARC and UltraSPARC-II.)
- Load/store architecture; Number of instructions is around 100 (RISC).
- Memory address = 64 bits; big endian; word size = 32 bits.
- Instruction length = 32 bits.
- 64-bit registers for integer operations.
- Has a "register file" consisting of 64 to 512 registers. Each procedure can use only a "window" of 32 registers.
- Windows of different procedures may overlap (to support parameter passing).

Sun SPARC Architecture (continued)

- Register file of floating point registers.
- Each floating point register is 32 bits long to provide for single precision IEEE FPS; registers are paired to support double precision.
- Fewer addressing modes compared to IA-32 (e.g. immediate, register indirect + displacement, register indirect + index, etc).
- Supports all basic data types of C.
- Separate floating point processor.