CMPE 220

Class 3

Computer Architecture

Simplified Instructional Computer

Simplified Instructional Computer (SIC)

- A hypothetical computer that includes the hardware features most often found on real machines.
  - SIC (standard model)
  - SIC/XE
- Upward compatible
  - Programs for SIC can run on SIC/XE
- SIC is a good example of the basic architectural features required in a computer.
- I'll use SIC for many of the examples in the remainder of this class

SIC - Memory

- 8-bit bytes
- 3 consecutive bytes form a 24-bit word
  - Words are addressed by the lowest number byte
- 2^{15} (32768) bytes in the computer memory

SIC - Registers

- Five 24-bit registers

Mnemonic	#	Use
A	0	**Accumulator**: used for arithmetic & logic operations
X	1	**Index**: used for addressing
L	2	**Linkage**: stores the return address for a subroutine jump. Only allows one level of return. SIC does not have a stack.
PC	8	**Program Counter**: contains the address of the next instruction to be fetched
SW	9	**Status Word**: see next page

SIC – Status Word (SW)

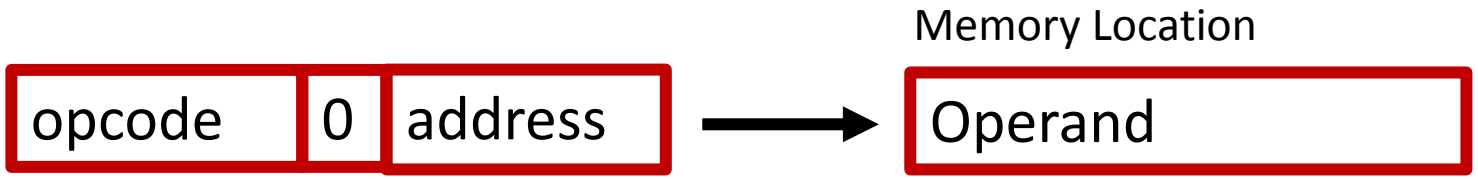
Field	Length (bits)	Bits	Use
mode	1	0	user mode (0) or supervising mode (1)
state	1	1	process is in running state (0) or idle state (1)
id	3	2-5	process id (PID)
CC	2	6-7	condition code (device state)
mask	4	8-11	interrupt mask
X	4	12-15	(unused)
icode	8	16-23	interrupt code i.e. Interrupt Service Routine

SIC – Instruction Format

- 24 bits (1 word)
- **opcode:** machine instruction code **(**8 bits)
- **m:** address mode (1 bit)
  - **0:** direct
  - **1:** indexed (aka *base register* addressing)
- **address**: target address, or offset from X register (15 bits)

SIC – Addressing Modes

- Direct



- Indexed (*Base Register* addressing)

SIC – Data Formats

- **Characters:** 8-bit ASCII
- **Integers:** 24-bit binary, two's complement
- No decimal, no floating point

SIC – Instructions (Load & Store)

- Transfer 1 word of data (24 bits) between the A, X, or L registers, and a memory location
  - LDA location
  - LDL location
  - LDX location
  - STA location
  - STL location
  - STX location
- Transfer 1 byte of data (8 bits) between the A register, and a memory location
  - LDCH location
  - STCH location

SIC – Instructions (Arithmetic & Logic)

- Arithmetic involves the A register and a memory location
  - ADD location
  - SUB location
  - MUL location
  - DIV location
  - AND location
  - OR location
- The contents of the A register are replaced by:
  A (operation) (contents of location)

SIC – Instructions (Comparison)

- Comparison instructions set the CC flag to <, =, or >
- Compare the A register and the contents of a memory location
  - COMP location
- Increment the X (index) register, and compare the result to the contents of a memory location
  - TIX location

SIC – Instructions (Jump)

- Unconditional Jump
  - J location
- Conditional Jumps (based on value of CC flag):
  - JEQ location
  - JLT location
  - LGT location
- Jump to subroutine; store return address in L register
  - JSUB location
- Return from subroutine (jump to address in L register)
  - RSUB

SIC – Input/Output

- Three I/O instructions
- **TD:** Test Device; returns status in CC flag of SW register. ‘<‘ means ready, ‘=‘ means not ready.
- **RD:** Read one byte of data from the specified device into the lower 8 bits of the A register.
- **WD:** Write one byte of data from the lower 8 bits of the A register to the specified device.

SIC – A Workable RISC Instruction Set

- Although lacking many “convenience” instructions, the SIC architecture implements a fully functional, general purpose instruction set, similar to common minicomputers of the 1960s.
- Its chief limitation is the 15-bit address space, allowing only 32,767 bytes of memory.

Non-Instruction Statements (SIC)

- So far, we've talked about statements that correspond one-to-one to machine code instructions… but there is another statement type required: the *memory declaration*.
- Reserve some memory, and assign a label:

```
inventory       RESW    5     (Reserve 5 words)
partnumbers     RESB    100   (Reserve 100 bytes)
```

- Reserve some memory, and assign a label *and starting value*:

```
inventory       WORD    100         (Reserve 1 word; value=100)
partname        BYTE    C'widget'   (Reserve 6 bytes; value='widget')
Iochannel       BYTE    X'05'       (Reserve 1 byte; value=x05)
```

Other Housekeeping Statements

- Indicated starting address of program:
  programname START 1000
- Indicate end of program, and location of first statement:
  END starthere

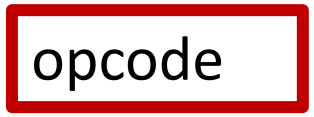
SIC/XE – An Extended CISC Instruction Set

- The SIC/XE is fully backward compatible with the SIC. That is, it will run all SIC instructions.
- It adds:
  - A 20-bit addressing mode, supporting 1 MB of memory
  - Floating point arithmetic
  - Multiple new addressing modes
  - Additional arithmetic and logic functions

SIC/XE – Additional Registers

Mnemonic	#	Use
B	3	**Base**: base register for addressing
S	4	**S**: general accumulator
T	5	**T**: general accumulator
F	6	**F**: floating-point accumulator (48 bits)

SIC/XE – Additional Instruction Formats



- 8-bit
- **opcode:** machine instruction code **(**8 bits)

- 16-bit
- **opcode:** machine instruction code **(**8 bits)
- **r1, r2**: register identifiers (4 bits, 4 bits)

SIC/XE – Additional Instruction Formats

- 24-bit
- **opcode:** machine instruction code **(**6 bits)
- **flags:** n, i, x, b, p, e (6 bits)
- **disp:** 12-bit address displacement

- 32-bit
- **opcode:** machine instruction code **(**8 bits)
- **flags:** n, i, x, b, p, e (6 bits)
- **address:** 20-bit address

SIC/XE – Instruction Flags

- **n**: Indirect addressing flag
- **i**: Immediate addressing flag
- **x**: Indexed addressing flag
- **b**: Base address-relative flag
- **p**: Program counter-relative flag
- **e**: Format 4 instruction flag

SIC/XE – Additional Addressing Modes

- Base Relative

SIC/XE – Additional Addressing Modes

- Program-Counter Relative

SIC/XE – Additional Data Formats

- Floating Point (48-bits)
  - **s:** 1-bit
  - **exponent:** 11-bits
  - **fraction:** 36-bits

SIC/XE – Instructions (Load & Store)

- Transfer 1 word of data (24 bits) between the B, S, or T registers, and a memory location
  - LDB location
  - LDS location
  - LDT location
  - STB location
  - STS location
  - STT location
- Transfer 2 words of data (48 bits) between the F register, and a memory location
  - LDF location
  - STF location

SIC/XE – Instructions (FP Arithmetic)

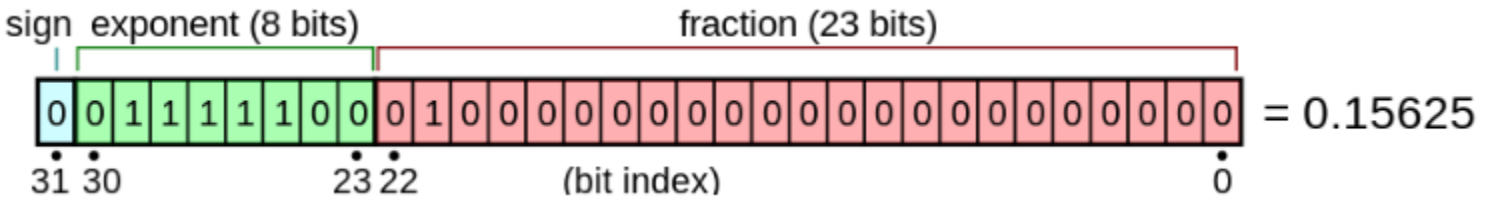
- Floating point arithmetic involves the F register and a memory location
  - ADDF location
  - SUBF location
  - MULF location
  - DIVF location
- The contents of the F register are replaced by:
  F (operation) (contents of location)

Precision and Portability: (an aside)

- In order for programs to be *portable* across platforms (hardware independent), arithmetic must work the same on any hardware.
- Language standards and test suites usually specify a *minimum* range and precision for each data type… but some systems support greater range and precision.
- A program may run correctly on a high-precision system, and fail when run on a lower precision system – even though both systems are standards-compliant. The programmers are inadvertently depending on higher precision than the standards.
- Some programs may even fail when running on *higher-precision* hardware.

Floating Point Arithmetic

- **IEEE 754:** a set of standards for binary and decimal floating point representation and arithmetic. The standard was established in 1985 and updated in 2008 and 2019.
- Each sub-standard specifies the number of bits (digits) and the exponent size.
  - **binary32:** 24 significant bits (including sign); 8 exponent bits



Floating Point Arithmetic - continued

- A floating point format consists of:
  - a base (also called *radix*) *b*, which is either 2 (binary) or 10 (decimal) in IEEE 754;
  - a precision *p*;
  - an exponent range from *emin* to *emax*, with *emin* = 1 − *emax* for all IEEE 754 formats
  - Specified representations for +infinity, -infinity, and NaN (not a number)
- The IEEE 754 floating-point standard also includes rules for rounding

SIC/XE – Instructions (Register Arithmetic)

- Perform an arithmetic operation on two specified registers:
  - **ADDR reg1, reg2:** reg2 = reg2 + reg1
  - **SUBR reg1, reg2:** reg2 = reg2 - reg1
  - **MULR reg1, reg2:** reg2 = reg2 * reg1
  - **DIVR reg1, reg2:** reg2 = reg2 / reg1
- Logical operations: circular shift the specified register ‘n’ bits
  - **SHIFTL r1,n:** circular shift left
  - **SHIFTR r1,n:** circular shift right
- **RMO reg1, reg2:** Move the first specified register to the second
- **CLEAR reg1:** Clear the specified register (set the value to 0)

SIC/XE – Instructions (Conversion)

- Convert the integer in the A register to floating point, and store the result in the F register
  - FLOAT
- Convert the floating in the F register to integer, and store the result in the A register
  - FIX
- Normalize the floating point number in the F register
  - NORM

SIC/XE – Instructions (Comparison)

- Comparison instructions set the CC flag to <, =, or >
- Compare the F register and the 48-bit contents of a memory location
  - COMPF location
- Compare the first specified register to the second specified register
  - COMPR reg1,reg2
- Increment the X (index) register, and compare the result to the contents of a specified register
  - TIXR register

Assembly Example: SIC/XE

Line #	Label	Instruction	Argument	*Address*	*Instruction Size**
1	Program	START	1000	*1000*	*0*
2		LDA	Inventory	*1000*	*3*
3		LDT	Sales	*1003*	*3*
4		SUBR	T, A	*1006*	*2*
5		J	DisplayRoutine	*1008*	*3*
6	Partnumber	BYTE	C'005740'	*1011*	*6*
7	Inventory	WORD	500	*1017*	*3*
8	Sales	WORD	27	*1020*	*3*

- Note that the SIC/XE has variable length instructions. This is often true of CISC machines. RISC machines have uniform length instructions.

Useful Resources

- Simplified Instructional Computer (SIC) Architecture
  - https://www.geeksforgeeks.org/simplified-instructional-computer-sic/
- SIC/XE Architecture
  - https://www.geeksforgeeks.org/sic-xe-architecture/
- SIC/XE Instruction Set
  - https://www.geeksforgeeks.org/instruction-set-used-in-sic-xe/
  - https://www.unf.edu/~cwinton/html/cop3601/supplements/test.html

The SIC/XE Simulator

- There is a Simulator for the SIC/XE machine
  - Integrated Development Environment
  - Assembler
  - Linker
  - Simulator (executes SIC/XE instructions)
- Download at: http://jurem.github.io/SicTools/
  - Written in Java
  - Download the JAR (Java Archive) file

Assignment 1

Due in One Week (at start of class)

- Log in to Canvas and complete Assignment 1
- Recommended: Download and install the SIC/XE simulator