EE 357 Unit 4a

Instruction Set Architecture

Instruction Set Architecture (ISA)

- Defines the software interface of the computer system
- Instruction set is the “vocabulary” that the HW can understand and that SW is composed with
- 2 approaches
  - CISC = Complex instruction set computer (Coldfire/M68K & Intel)
    - Large, rich vocabulary
    - More work per instruction but slower HW
  - RISC = Reduced instruction set computer (MIPS / PPC / ARM)
    - Small, basic, but *sufficient* vocabulary
    - Less work per instruction but faster HW

RISC & CISC Comparison

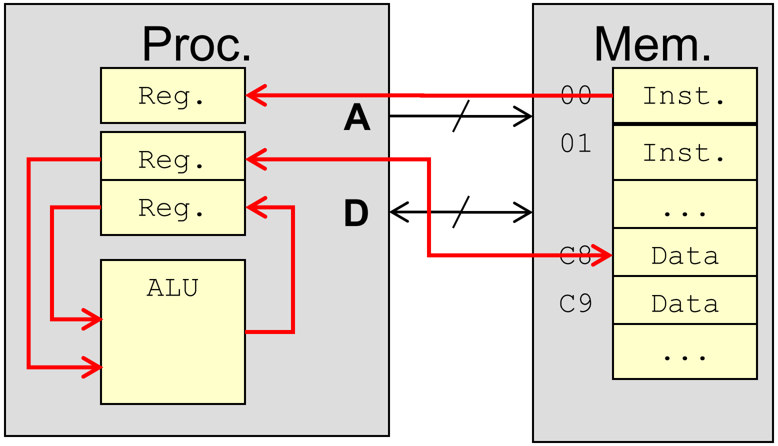
- Which is better?
  - Exec. Time of a Program = Time to do all operations/work

$$Time = \frac{TotalWork}{} \bullet \frac{1}{\frac{Work}{Instruc}} \bullet \frac{Time}{Instruc}$$

  - With CISC: (Work/Instruc.) = ↑ while (Time / Instruc.) ↑
  - With RISC: (Work/Instruc) = ↓ while (Time / Instruc.) ↓
- One is not inherently better, though most computers at the HW level implement RISC-style instructions because fewer operations per instruction makes the HW easier to design.

Computer Organization Overview

- A processor has to…
  - Read instructions from memory
  - Read necessary data into registers
    - Fast storage locations inside the processor used to store values as the processor operates on them
  - Perform operations on the data
  - Write data back to memory or I/O device



Components of an ISA

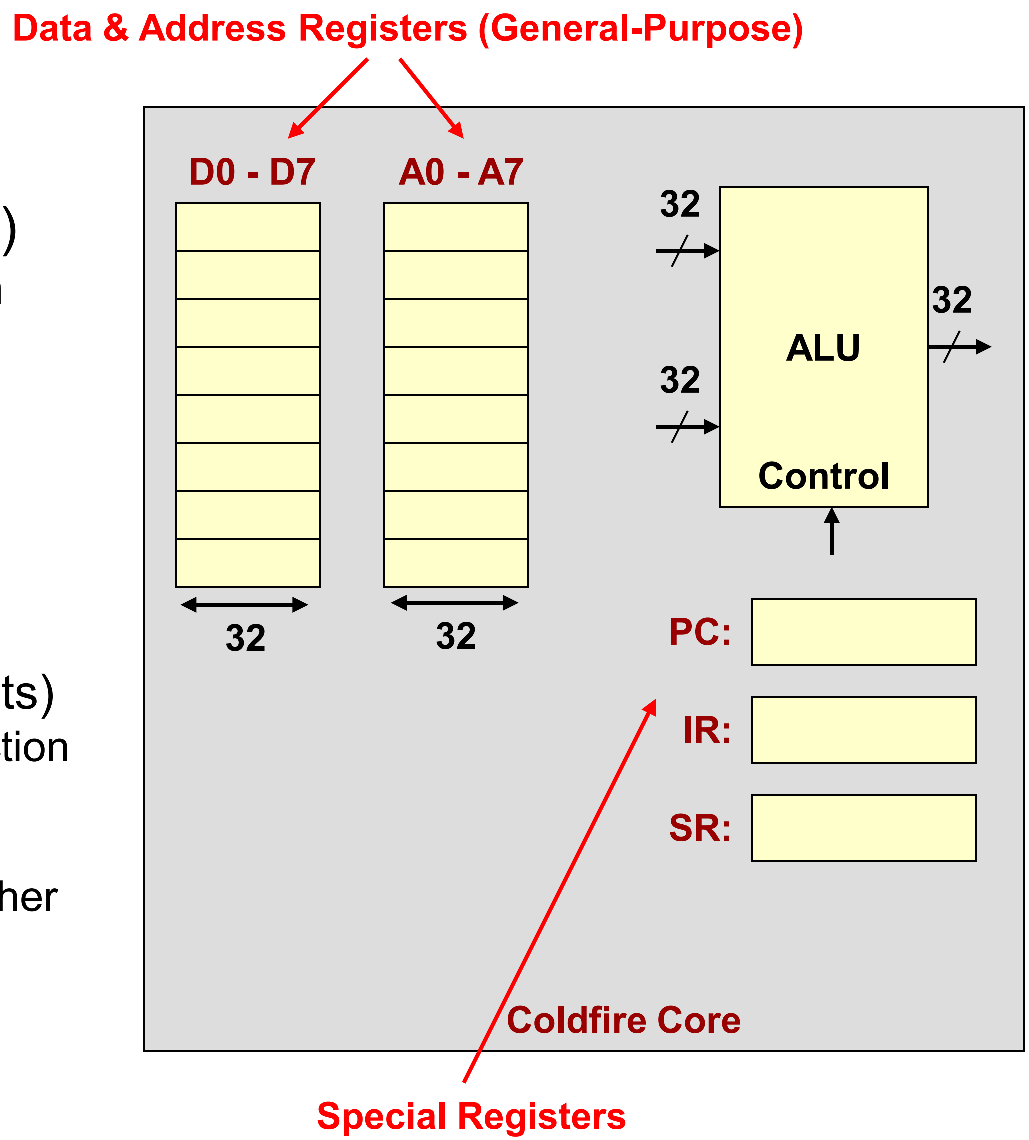
1. Data and Address Size
   – 8-, 16-, 32-, 64-bit
2. Which instructions does the processor support
   – SUBtract instruc. vs. NEGate + ADD instrucs.
3. Registers accessible to the instructions
4. Addressing Modes
   – How instructions can specify location of data operands
5. Length and format of instructions
   – How is the operation and operands represented with 1's and 0's

Coldfire/M68K ISA

- CISC style
- 32-bit data and address
  - # of address bits determines max amount of memory
  - Since addresses are also 32-bits => Max 4GB of memory
- Separate data and address registers
  - 8 data registers (D0-D7)
  - 8 address registers (A0-A7)
    - Intended to store pointers (address to other data)
    - Can be used for data
- Instructions = One 16-bit instruction word followed by 0-2 extension words stored after the instruction word
  - Instruction word stores all information about operation and how to find the operands, while extension words are used depending on the specific instruction and store specific information used by the instruction

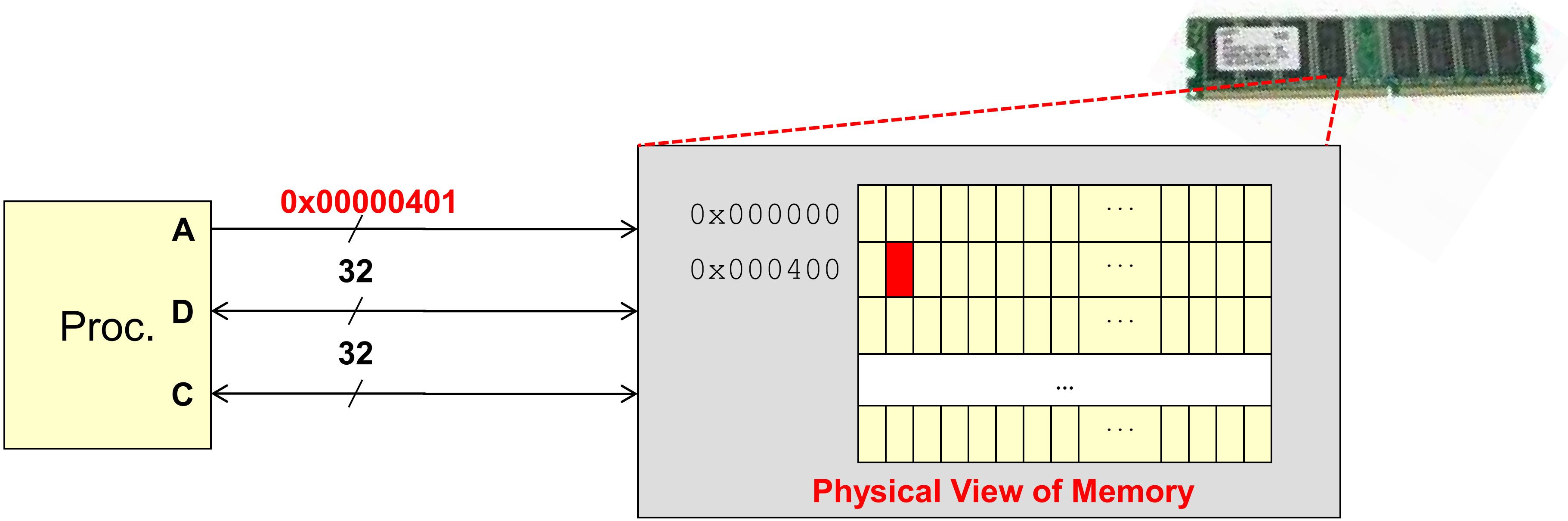
Coldfire Processor Core

- Data Registers (32-bits)
  - Hold data operands
- Address Registers (32-bits)
  - Act like pointers to data in memory
- Special Registers
  - PC: Program Counter (32-bits)
    - Holds the address of the next instruction to be executed
  - IR: Instruction Reg. (16-bits)
    - Holds the current instruction as it executes
  - SR: Status Reg. (16-bits)
    - Holds control bits and other "state" of the processor



Physical View of Main Memory

- Physical view of memory as large 2-D array of bytes
  - (e.g. 8K rows by 1KB columns) per chip (and several chips)
- Most computer memory is byte-addressable
  - Each byte has a unique address
- Processor interfaces via address, data, and control buses



Coldfire Memory Interface

- Address bus = 32-bits
- Data bus = 32-bits
  - Can read/write up to 4-bytes at once
- Memory can be read or written in 3 sizes
  - 1-byte (.B)
  - 2-bytes = 1 word (.W)
  - 4-bytes = 1 longword (.L)
  - Always specify the start address & size and the memory will take the appropriate sequential bytes
- Valid words start @ addresses that is multiples of 2
- Valid longwords start @ addresses that are multiples of 4

Address				
0x000	5A	13	F8	7C
0x004	8E	29	B4	36

Logical View of Memory Organized as 4-byte (1 Longword) rows

Longword @ Addr. 0

	Word @ Addr. 0		Word @ Addr. 2	
0x000	5A	13	F8	7C
	Byte @ 0	Byte @ 1	Byte @ 2	Byte @ 3

Longword @ Addr. 4

	Word @ Addr. 4		Word @ Addr. 6	
0x004	8E	29	B4	36
	Byte @ 4	Byte @ 5	Byte @ 6	Byte @ 7

Instruction Format

- Instructions must specify three things:
  - Operation (OpCode)
  - Source operands & where to find them (reg. or mem.)
    - Usually 2 source operands (e.g. X+Y)
  - Destination Location (reg. or mem.)
- Example: ADD D0,D1,D2 (D0=D1+D2)
- Binary (machine-code) representation broken into fields of bits for each part
- Some instruction sets used
  - Fixed-size instructions (MIPS): all instruction encoded with same # of bits
  - Variable-size instructions (Coldfire, Intel): differing lengths for different instructions

	OpCode	Dest.	Src. 1	Src. 2
Fictitious Example:	1101	1000	1001	1010
	ADD	D0	D1	D2

Instruction Format

- Different instruction sets specify these differently
  - 3 operand instruction set (MIPS)
    - Similar to example on previous page
    - Format: **ADD DST,SRC1,SRC2** (DST = SRC1 + SRC2)
  - 2 operand instructions (Coldfire/M68K & Intel/AMD)
    - Second operand doubles as source and destination
    - Format: **ADD SRC1,S2/D** (S2/D = SRC1 + S2/D)
  - 1 operand instructions (Some low-end embedded)
    - Implicit operand to every instruction usually known as the Accumulator (or ACC) register
    - Format: **ADD SRC1** (ACC = ACC + SRC1)

Instruction Format

- Consider the pros and cons of each format when performing the set of operations
  - F = X + Y – Z
  - G = A + B
- Embedded computers often use single operand format
  - Smaller data size (8-bit or 16-bit machines) means limited instruc. size
- Today's computers use 2- and 3-operand formats

Single-Operand	Two-Operand	Three-Operand

LOAD X
ADD Y
SUB Z
STORE F
LOAD A
ADD B
STORE G
	MOVE X,F
	ADD Y,F
	SUB Z,F
	MOVE A,G
	ADD B,G
		ADD F,X,Y
		SUB F,F,Z
		ADD G,A,B

(+) Smaller size to encode each instruction (-) Higher instruction count to load and store ACC value	Compromise of other two	(+) More natural program style (+) Smaller instruction count (-) Larger size to encode each instruction