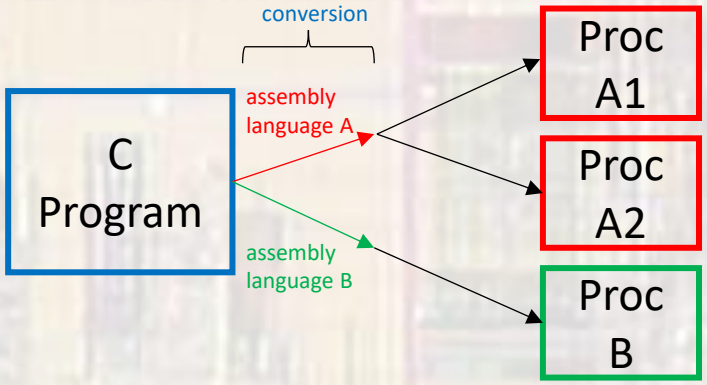
Assembly Language

Last updated 10/15/26

Assembly Language

- Assembly Language
  - An intermediate programming language
    - Between C and Machine Language (covered elsewhere)
  - Directly manipulates the elements of the processor
    - References registers, the ALU, and the memories
    - Processors of the same architecture can share a single assembly language since they share the same elements
    - Processors with different architectures will have different assembly languages since each processor architecture has different elements



© tj

Assembly Language

- Pretend processor architecture
  - Harvard Architecture
  - RISC – Load/Store Instruction Set
  - 16 bit instruction words
  - 4 – 8 bit data registers available for executing instructions (A-B-C-D)
  - Support for 16 instructions
  - 3 – memory based instructions

© tj

Assembly Language

- Arithmetic Instruction Format
  - Perform arithmetic operations on registers
  - General format:

    Instruction source-1, source-2, destination

    where source-1, source-2, and destination are registers A-D

code	operation
fn Reg1, Reg2, Wreg	Wreg ← Reg1 fn Reg2

  - Examples
    - add RA, RB, RC — RC ← RA + RB
    - or RB, RD, RB — RB ← RB or RD
    - sub RC, RA, RD — RD ← RC – RA
    - slt RC, RA — ALUout ← 1 when RC < RA; ALUout ← 0 when RC >= RA

A	0x12
B	0x23
C	0x35
D	0xF5

A	0x12
B	0xF7
C	0x35
D	0xF5

A	0x12
B	0xF7
C	0x35
D	0x23

© tj

Assembly Language

- Memory Instruction Format
  - Perform memory operations on registers
  - General format:

    Instruction address_register, memory_register

	code	operation
load	ld Reg1, Wreg	Wreg ← MEM(Reg1)
store	st Reg1, Reg2	MEM(Reg1) ← Reg2

  - Examples
    - ld RA, RC

      RC ← MEM(RA)

      RC ← MEM(0x02) = 0x44

    - st RD, RB

      MEM(RD) ← RB

      MEM(0xFE) ← 0x23

A	0x02
B	0x23
C	0x44
D	0xFE

0xFF	0xC3
0xFE	0x23
…	
0x02	0x44
0x01	
0x00	0x2B

© tj

Assembly Language

- Load Immediate Instruction Format
  - Directly load a value to a register
  - General format:

    Instruction destination_register, value

code	operation
ldi Wreg, “imm value”	Wreg ← “imm value”

A	0x12
B	0x23
C	0x44
D	0xFE

  - Examples
    - ldi RA, 0x12 RA ← 0x12
    - Note: the immediate value is actually part of the instruction

© tj

Assembly Language

- Branch Instruction Format
  - Modify what the next instruction will be
    - PC (program counter) points to the next instruction to execute

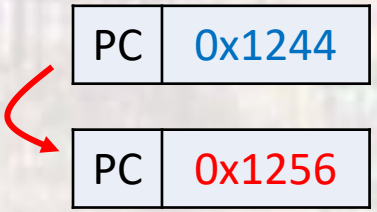
A	0x20
B	0x23
C	0x20
D	0xFE

  - BRE - Branch if Equal

    Instruction reg1, reg2, branch_distance

code	operation
bre Reg1, Reg2, offset	PC ← PC + offset if Reg1 = Reg2

  - Examples
    - bre RA, RC, 0x12 PC ← PC + 0x12 if RA = RC



 © tj

Assembly Language

- SLT – Set on Less than
  - Determine if A < B
    - A < B → (A – B) < 0 → negative answer
      - In hardware, subtraction is implemented by addition
      - A – B → A + $\overline{B}$ + 1
    - MSB indicates sign in 2's complement arithmetic
      - MSB = 1 → negative number → A < B
      - MSB = 0 → positive number → A >= B
  - Set On Less Than instruction
    - Use Adder MSB for SLT signal
    - If A < B: Out[31:1] = 0, Out[0] = 1
    - If A ≥ B: Out[31:0] = 0, Out[0] = 0
    - SET = 1

© tj