Instruction sets
MIPS assembly language

Decision-making operations

- High-level languages: if-else, for, while, etc.
- Instruction set: can be accomplished with a combination of the following:
  - branch on comparison
  - goto (“jump”)
  - labels

MIPS decision-making instructions

- beq: branch if equal
- bne: branch if not equal
- j: jump (unconditional)
- slt/slti: set on less than {register, immediate}
  - compares the contents of 2 registers; 3rd register is set to 1 if first register is less than second register, 0 otherwise
- jump address table: list of addresses for various program possibilities
  - switch statements

Example: for loop

```c
int prod = 3;
for (int i=1; i<10; i++) {
    prod = prod+i;
}
```

Q: What does this look like in MIPS assembly?
MIPS for loop (one solution)

li $s0, 3  # "load immediate"
li $s1, 1  # (pseudoinstruction)
li $t1, 1

Loop:  add $s0, $s0, $s1
       addi $s1, $s1, 1
       slti $t0, $s1, 10
       beq $t0, $zero, End
       j Loop

End:

Pseudoinstructions

- “Fake” instructions that are treated as if they are actual instructions by the assembler
- Not implemented in hardware
- Typically a “simpler” version of the actual instruction
- Examples:
  - move $s0, $s1  --> add $s0, $zero, $s1
  - blt $s0, $s1, Here -->
    slt $t0, $s0, $s1
    bne $t0, $zero, Here

Zero register

- $zero
- Because comparisons to 0 are so commonly used, there's a special register set aside to just store 0

Note

- All comparisons can be accomplished by some combination of slt, slti, beq, bne, and $zero
Procedures/functions/methods

- Perform a specific task (often times repetitively)
- Have “temporary” data associated with them
- Require passing of data (and control) between the calling program and the procedure itself

Procedures in MIPS: data

- Four registers for arguments (from the main program to the procedure)
  - $a0-$a3
- Two registers for return values (from the procedure to the main program)
  - $v0, $v1
- One register for the “return address” (where to return control in the main program) --> $ra

Procedures in MIPS: data (cont)

- Q: What registers can we use for the variables within the procedure?
- A:
  - $t0-$t7 can be used as-is
    - value not preserved upon return
    - caller responsible for storing values before calling the procedure
  - $s0-$s7: must copy these values somewhere else and reset them before returning
    - procedure’s responsibility to do so

Procedures in MIPS: data

- Stack
  - for (temporarily) storing data elsewhere
  - stack pointer register ($sp) stores the location of the “top” (last) item on the stack
  - add items: subtract from stack pointer
  - remove items: add to stack pointer