Lecture 4

CMSC 350: COMPILER DESIGN
The target architecture for CMSC 350

HERA
HERA Machine State: Registers

- Register File: 16 16-bit registers
  - R0           always 0
  - R1–R10      general purpose registers
  - R11/Rt      temporary register
  - R12/FP_alt  temporary frame pointer
  - R13/PC_ret  program counter return value
  - R14/FP      frame pointer; points to current stack frame
  - R15/SP      stack pointer; points to top of stack

- PC          a “virtual” register, points to the current instruction
  - PC is manipulated only indirectly via jumps and return.

- Flags, directly manipulated via special instructions, set by many as side effect
  - F0/s        whether last operation resulted in a negative number
  - F1/z        whether last operation resulted in 0
  - there are others; we'll ignore for now
Simplest instructions: SETLO, SETHI

- **SETLO**\((d, v)\) \( R_d \leftarrow v \)  
  Set a register \((v \text{ is 8 bits})\)

- **SETHI**\((d, v)\) \( (R_d)_{15:8} \leftarrow v \)  
  Set the upper byte of a register

Here, \(d\) and \(v\) are **operands**

- **\(d\)** is a **register**
- **\(v\)** is treated as a **value** (also called an **immediate**)

- **SETLO**\((R3, 0x12)\) changes the lower byte of \(R3\)
- **SETHI**\((R3, 0x34)\) changes the upper byte of \(R3\)
- After these two commands, \(R3\) will be \(0x3412\)
Arithmetic instructions

- **AND**(d,a,b) \[ R_d \leftarrow R_a \land R_b \]
- **OR**(d,a,b) \[ R_d \leftarrow R_a \lor R_b \]
- **XOR**(d,a,b) \[ R_d \leftarrow R_a \oplus R_b \]
- **ADD**(d,a,b) \[ R_d \leftarrow R_a + R_b \]
- **SUB**(d,a,b) \[ R_d \leftarrow R_a - R_b \]
- **MUL**(d,a,b) \[ R_d \leftarrow R_a \times R_b \]

- d, a, and b are all registers
- These operations set the s and z flags: s is set if the result is negative (if the highest bit is set) and z if the result is 0.
HERA State: Condition Flags & Codes

- HERA instructions set flags as a side effect
- HERA has 5 flags:
  - s: “signed” set when the result is negative (e.g., when the top bit is set -- see "two's-complement binary encoding")
  - z: “zero” set when the result is 0
  - v: “overflow” set when the result of an operation cannot fit in 16 bits
  - c: “carry” set when an operation must carry or not borrow
  - cb: “carry block” set when the programmer wishes not to worry about carrying

- We will not worry about the last three in this course.
- **Exception:** HERA-C uses the c flag when doing arithmetic. Disable this by using CBON( ), which sets the cb flag, disabling the c flag.
HERA Branches

- BR(lbl)  unconditional branch to lbl
- BZ(lbl)  branch to lbl if z flag is set
- BL(lbl)  branch to lbl if s flag is set (L = "Less than 0")
- many others

HERA supports both relative and absolute branches
- A relative branch gives an 8-bit signed value, saying by how much to change the PC
- An absolute branch gives a register number (4 bits), where that register says what the new PC should be
- Difference between these is immaterial in HERA-C, but is very important in real code (and HW02)
- Not all branches can be relative, because 8 bits might not be enough to get you to where you are going. (Recall that HERA addresses are 16 bits.)
The HERA memory consists of $2^{16}$ 16-bit (2-byte) words, numbered $0x0000$ through $0xffff$. (Thus, there are $2^{17}$ bytes in the HERA memory.)

An address is treated as a word address. There is no way to address an individual byte.

Program text (i.e., instructions) is located in a different memory structure. HERA does not support reading or modifying program text other than by executing instructions.

By convention:
- the stack starts at address $0x0000$ and grows upward.
- the heap starts at address $0x4000$ and grows upward.
- the global data starts at address $0xc000$ and grows upward.
HERA memory access

- \text{LOAD}(d, off, b) \quad R_d \leftarrow M[R_b + off]
- \text{STORE}(d, off, b) \quad M[R_b + off] \leftarrow R_d

- In both instructions, \( R_b \) is a base address. A 5-bit unsigned offset \( off \) is added to this base address to get the memory address.
- The use of \( off \) gives efficient access to arrays containing up to 32 elements.
Code Blocks & Labels

• HERA assembly code is organized into labeled blocks:

```
LABEL(label1)
    <instruction>
    <instruction>
    ...
    <instruction>

LABEL(label2)
    <instruction>
    <instruction>
    ...
    <instruction>
```

• Labels indicate code locations that can be jump targets (either through conditional branch instructions or function calls).
• Labels are translated away by the linker and loader
• A HERA program begins executing at instruction 0x0000. By convention, this block is labeled main.
Function calls

- HERA stack grows up from the bottom
- Described in Section 7.5 of HERA booklet
- Suppose `main` has called `f`, and we are currently executing `f`. Then, our stack looks like the diagram.
- Each frame stores (in order of ascending address):
  - Return address (where to continue executing after we're done)
  - Previous frame pointer (bottom of previous stack frame)
  - Static link (more about this in several weeks)
  - Function parameters
  - Local storage
- The `FP` register stores the bottom of the frame
- The `SP` register stores the top
FUNCTION CALL EXAMPLE