Today's Lecture

1) System lab?
2) C -> Assembly / Processors
3) RISC-V & x86
4) Registers
5) Stack & Calling Conventions
6) Struct Layout in Memory.
C → Asm
int main ( ) { print; 
exit();
}

ISA → Instruction set
C → Asm → Binary (Object/ .o files)
( .S files)
add, mul, etc ...

C++
RISC-V vs. x86-64

RISC-V

ISA → Personal Computers

x86-64

ISA → Complex ISA

CISC

• 3 full books
• 3 inst/month (15k Instr)

ARM (RISC)

• Qualcomm Snapdragon (Android)
• iOS (Apple)

RISC-V → Integrated devices

• Fewer Inst
• Simple Instr
• Open Source
<table>
<thead>
<tr>
<th>Register</th>
<th>ABI Name</th>
<th>Description</th>
<th>Saver</th>
</tr>
</thead>
<tbody>
<tr>
<td>x0</td>
<td>zero</td>
<td>Hard-wired zero</td>
<td>—</td>
</tr>
<tr>
<td>x1</td>
<td>ra</td>
<td>Return address</td>
<td>Caller</td>
</tr>
<tr>
<td>x2</td>
<td>sp</td>
<td>Stack pointer</td>
<td>Callee</td>
</tr>
<tr>
<td>x3</td>
<td>gp</td>
<td>Global pointer</td>
<td>—</td>
</tr>
<tr>
<td>x4</td>
<td>tp</td>
<td>Thread pointer</td>
<td>—</td>
</tr>
<tr>
<td>x5–7</td>
<td>t0–2</td>
<td>Temporaries</td>
<td>Caller</td>
</tr>
<tr>
<td>x8</td>
<td>s0/fp</td>
<td>Saved register/frame pointer</td>
<td>Callee</td>
</tr>
<tr>
<td>x9</td>
<td>s1</td>
<td>Saved register</td>
<td>Callee</td>
</tr>
<tr>
<td>x10–11</td>
<td>a0–1</td>
<td>Function arguments/return values</td>
<td>Caller</td>
</tr>
<tr>
<td>x12–17</td>
<td>a2–7</td>
<td>Function arguments</td>
<td>Caller</td>
</tr>
<tr>
<td>x18–27</td>
<td>s2–11</td>
<td>Saved registers</td>
<td>Callee</td>
</tr>
<tr>
<td>x28–31</td>
<td>t3–6</td>
<td>Temporaries</td>
<td>Caller</td>
</tr>
<tr>
<td>f0–7</td>
<td>ft0–7</td>
<td>FP temporaries</td>
<td>Caller</td>
</tr>
<tr>
<td>f8–9</td>
<td>fs0–1</td>
<td>FP saved registers</td>
<td>Callee</td>
</tr>
<tr>
<td>f10–11</td>
<td>fa0–1</td>
<td>FP arguments/return values</td>
<td>Caller</td>
</tr>
<tr>
<td>f12–17</td>
<td>fa2–7</td>
<td>FP arguments</td>
<td>Caller</td>
</tr>
<tr>
<td>f18–27</td>
<td>fs2–11</td>
<td>FP saved registers</td>
<td>Callee</td>
</tr>
<tr>
<td>f28–31</td>
<td>ft8–11</td>
<td>FP temporaries</td>
<td>Caller</td>
</tr>
</tbody>
</table>

**load value → reg**

**operate on reg**

**store reg →**

**Caller** → Not preserved across fn call

**Callee** → preserved across fn call
the Stack

Stack frame (generated by fn calls)

fp ⇒ top of current frame

Function prologue

Body

Epilogue

Return Address
To Prev. Frame (fp)
Saved Registers
Local Variables
...

Return Address
To Prev. Frame (fp)
Saved Registers
Local Variables
...

Asm Function

Low

fp

sp

HIGH
Figure 3.4: A process’s user address space, with its initial stack.