Today's Lecture

1) Syscall lab?
2) C -> ASM / Processors
3) RISC-V & X86
4) Registers
5) Stack + Calling Conventions
6) Struct Layout in Memory.

C → Asm

```
int main ( ) {   Print;
                 exit();
}
```

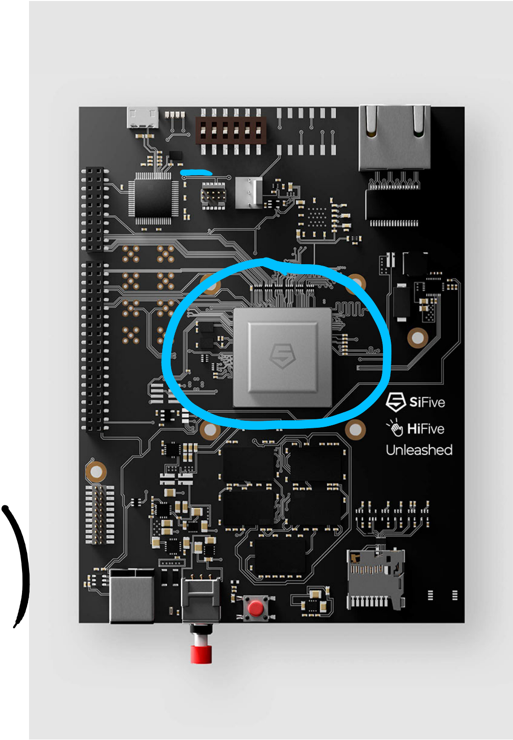
ISA → Instruction Sets

C → Asm → Binary (Object / .o files)

(.S files)

add, mult, etc ...

C++



RISC-V vs. x86-64

└> Reduced ISA

└> ISA → Personal computers (Intel, AMD)

CISC → Complex ISA

x86-64

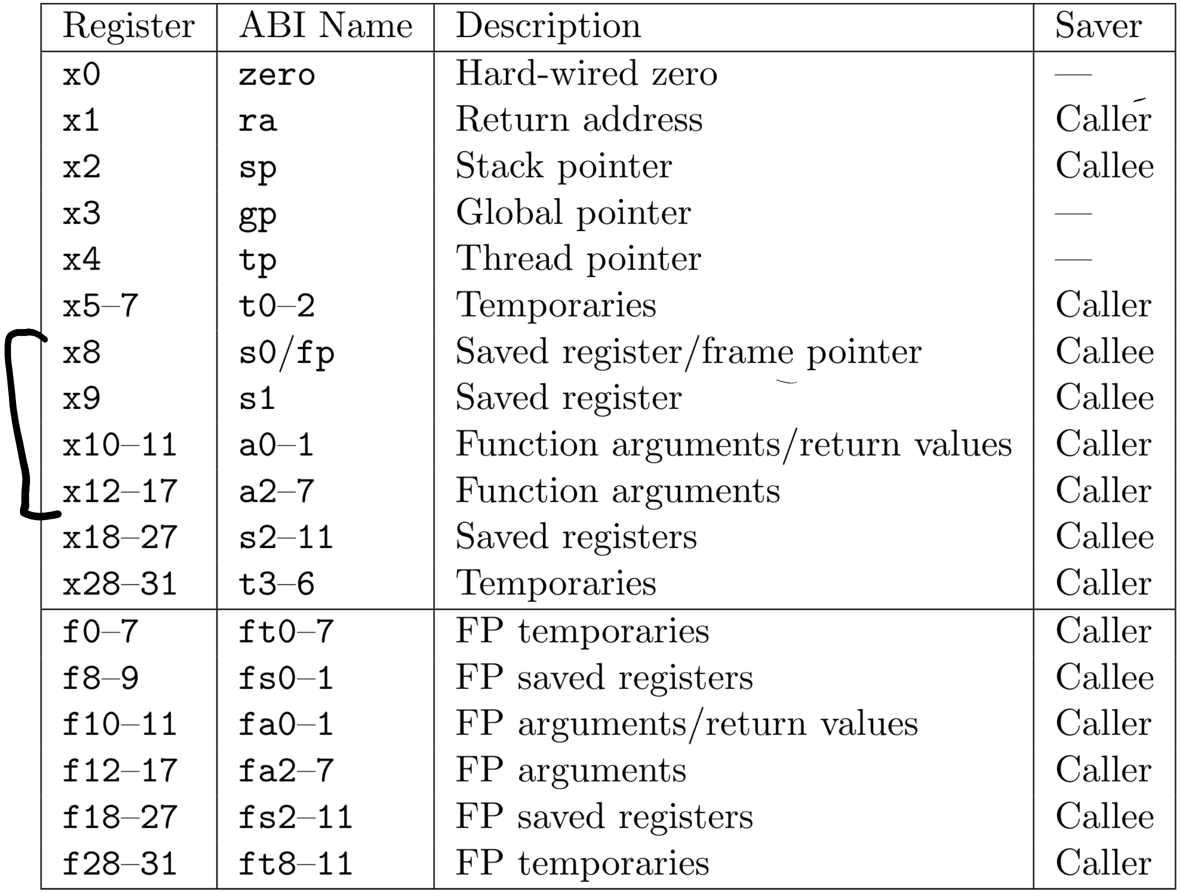
- 3 full books
- 3 inst/month (15k Instr)

RISC-V

- Fewer Inst
- Simple Instr
- Open Source

- ARM (RISC)
  - Qualcomm snapdragon (Android)
  - IOS (Apple)

RISC-V → Integrated devices



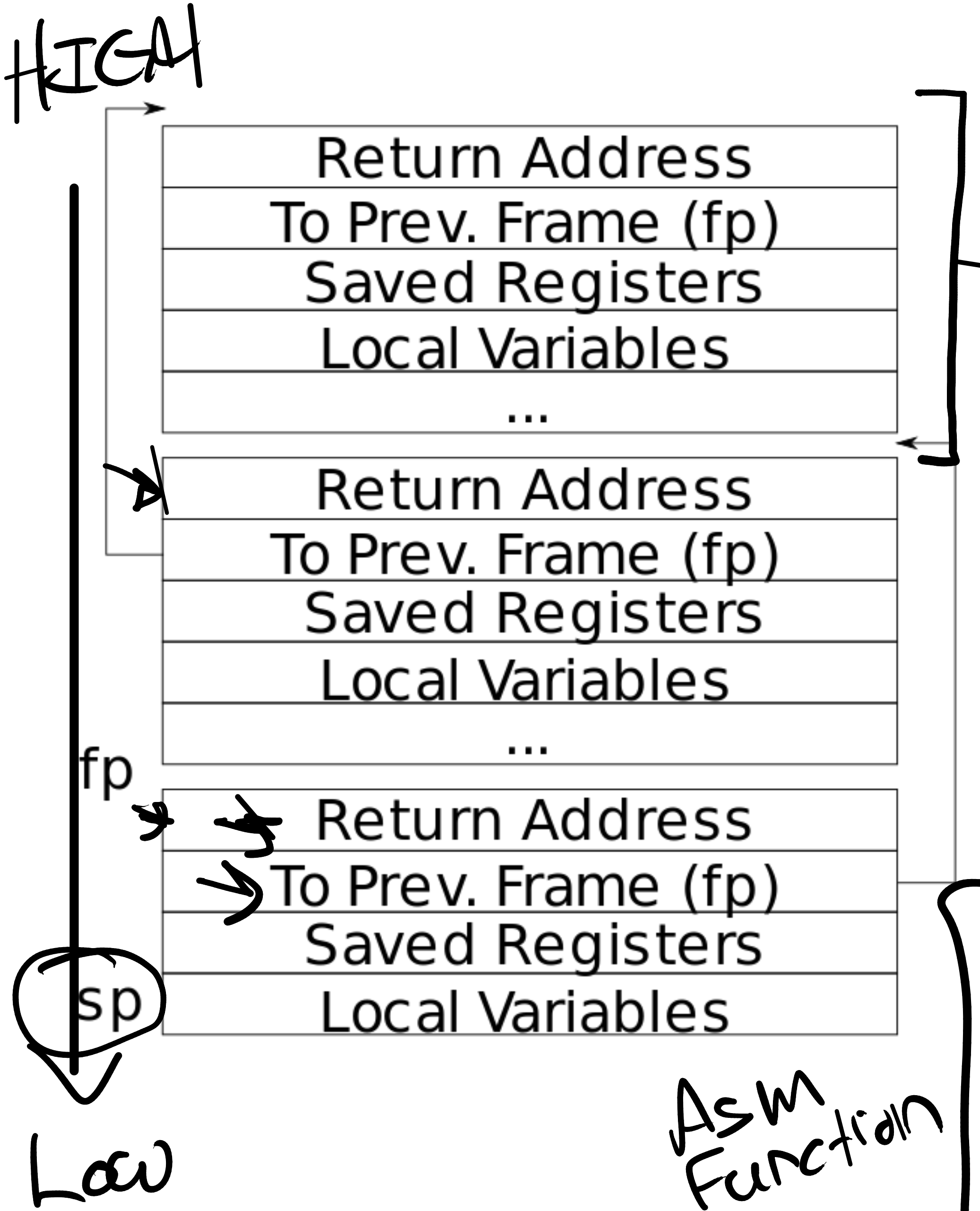
Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5–7	t0–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return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

load value -> reg
operate on reg
Store reg ->

Caller -> Not preserved across fn call

callee -> preserved across fn call

the stack



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

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

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

stack frame
(Generated by fn calls)

SP -> Bottom of stack?

fp => top of current frame

Asm Function

Function prologue
Body
Epilogue

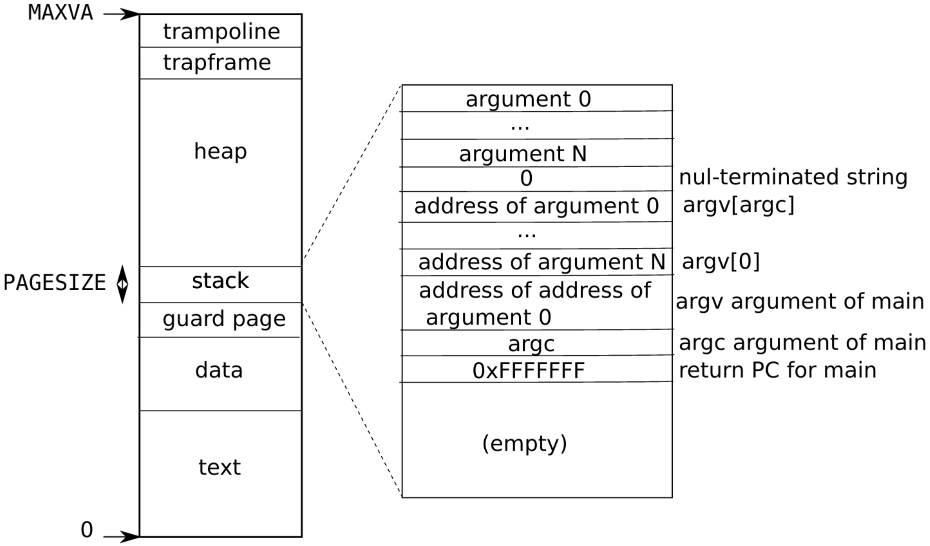


Figure 3.4: A process's user address space, with its initial stack.

Struct{
  f1
  f2
  f3
}