Computer Architecture

Week 10: RISC, CISC, ISA

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

Email: ugur.ozbalkan@fbu.edu.tr

Course Plan

- RISC
- CISC
- ISA

Big Picture: Where are we going?

C

compiler

RISC-V assembly

assembler

machine code

CPU

Circuits

Gates

Transistors

Silicon

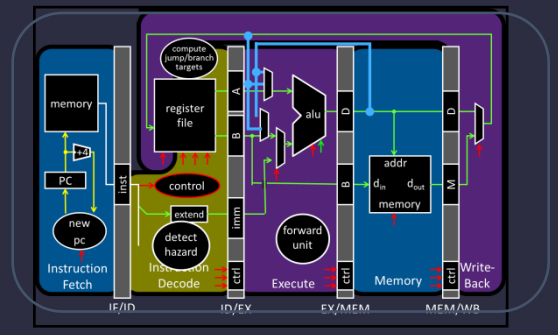
```
int x = 10;
x = 2 * x + 15;
```

High Level Languages

```
addi x5, x0, 10
muli x5, x5, 2
addi x5, x5, 15
```

```
00000000101000000000001010010011
00000000001000101000001010000000
00000000111100101000001010010011
```

Instruction Set Architecture (ISA)



Goals for Today

Instruction Set Architectures

- ISA Variations, and CISC vs RISC
- Peek inside some other ISAs:
  - X86
  - ARM

Next Goal

Is RISC-V the only possible instruction set architecture (ISA)?

What are the alternatives?

Instruction Set Architecture Variations

ISA defines the instructions

- RISC-V: load/store, arithmetic, control flow, …
- ARMv7: similar to RISC-V, but more shift, memory, & conditional ops
- ARMv8 (64-bit): even closer to RISC-V, no conditional ops
- VAX: arithmetic on memory or registers, strings, polynomial evaluation, stacks/queues, …
- x86: a little of everything

In the Beginning…

People programmed in assembly and machine code!

- Needed as many addressing modes as possible
- Memory was (and still is) slow

CPUs had relatively few registers

- Register's were more "expensive" than external mem
- Large number of registers requires many bits to index

Memories were small

- Encouraged highly encoded microcodes as instructions
- Variable length instructions, load/store, conditions, etc

In the Beginning…

People programmed in assembly and machine code!

E.g. x86

- > 1000 instructions!
  - 1 to 15 bytes each
  - E.g. dozens of add instructions
- operands in dedicated registers, general purpose registers, memory, on stack, …
  - can be 1, 2, 4, 8 bytes, signed or unsigned
- 10s of addressing modes
  - e.g. Mem[segment + reg + reg*scale + offset]

E.g. VAX

- Like x86, arithmetic on memory or registers, but also on strings, polynomial evaluation, stacks/queues, …

Reduced Instruction Set Computer (RISC)

RISC-V Design Principles

Simplicity favors regularity
- 32 bit instructions
- Same instruction format works at 16- or 64-bit formats

Smaller is faster
- Small register file

Make the common case fast
- Include support for constants

Good design demands good compromises
- Support for different type of interpretations/classes

Reduced Instruction Set Computer

RISC-V = Reduced Instruction Set Computer (RISC)

- ≈ 200 instructions, 32 bits each, 4 formats
- all operands in registers
  - 32 bits each
- ≈ 1 addressing mode: Mem[reg + imm]

x86 = Complex Instruction Set Computer (CISC)

- > 1000 instructions, 1 to 15 bytes each
- operands in dedicated registers, general purpose registers, memory, on stack, …
  - can be 1, 2, 4, 8 bytes, signed or unsigned
- 10s of addressing modes
  - e.g. Mem[segment + reg + reg*scale + offset]

The RISC Tenets

RISC

- Single-cycle execution
- Hardwired control

- Load/store architecture
- Few memory addressing modes
- Fixed-length insn format

- Reliance on compiler optimizations
- Many registers (compilers are better at using them)

CISC

- many multicycle operations
- microcoded multi-cycle operations
- register-mem and mem-mem
- many modes

- many formats and lengths

- hand assemble to get good performance
- few registers

RISC vs CISC

RISC Philosophy

Regularity & simplicity

Leaner means faster

Optimize the common case

Energy efficiency

Embedded Systems

Phones/Tablets

CISC Rebuttal

Compilers can be smart

Transistors are plentiful

Legacy is important

Code size counts

Micro-code!

Desktops/Servers

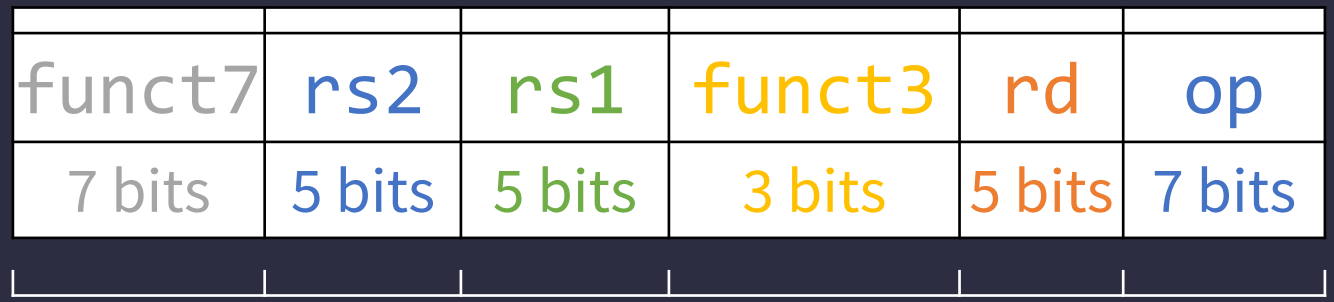
Next Goal

How does RISC-V and ARM compare to each other?

RISC-V instruction formats

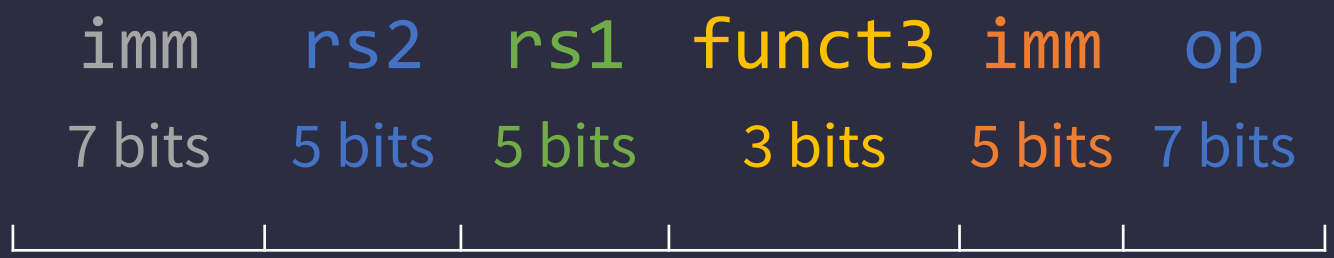
All RISC-V instructions are 32 bits long, have 4 formats

- R-type



- I-type

- S-type



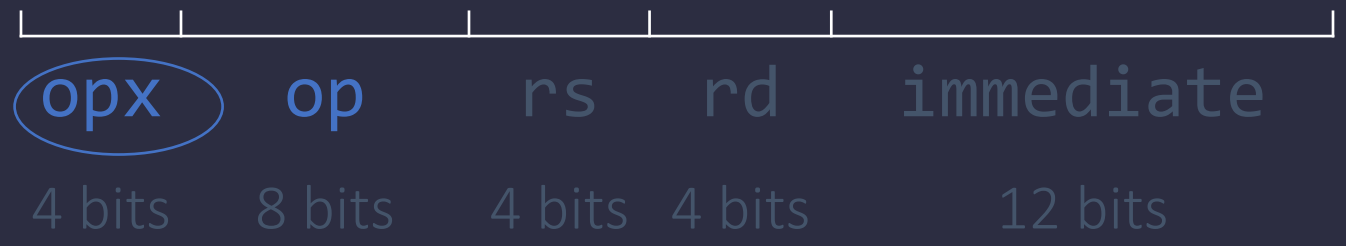
- U-type

ARMv7 instruction formats

All ARMv7 instructions are 32 bits long, has 3 formats

R-type

I-type



J-type

ARMv7: Other Cool operations

Shift one register (e.g. Rc) any amount

Add to another register (e.g. Rb)

Store result in a different register (e.g. Ra)

ADD Ra, Rb, Rc LSL #4

Ra = Rb + Rc<<4

Ra = Rb + Rc x 16

ARMv7 Instruction Set Architecture

All ARMv7 instructions are 32 bits long, has 3 formats

Reduced Instruction Set Computer (RISC) properties

- Only Load/Store instructions access memory
- Instructions operate on operands in processor registers
- 16 registers

Complex Instruction Set Computer (CISC) properties

- Autoincrement, autodecrement, PC-relative addressing
- Conditional execution
- Multiple words can be accessed from memory with a single instruction (SIMD: single instr multiple data)

ARMv8 (64-bit) Instruction Set Architecture

All ARMv8 instructions are 64 bits long, has 3 formats

Reduced Instruction Set Computer (RISC) properties

- Only Load/Store instructions access memory
- Instructions operate on operands in processor registers
- **32** registers and r0 is always 0

NO MORE Complex Instruction Set Computer (CISC) properties

- NO Conditional execution
- NO Multiple words can be accessed from memory with a single instruction (SIMD: single instr multiple data)

ISA Takeaways

The number of available registers greatly influenced the instruction set architecture (ISA)

Complex Instruction Set Computers were very complex

+ Small # of insns necessary to fit program into memory.

- greatly increased the complexity of the ISA as well.

Back in the day… CISC was necessary because everybody programmed in assembly and machine code! Today, CISC ISA's are still dominant due to the prevalence of x86 ISA processors. However, RISC ISA's today such as ARM have an ever increasing market share (of our everyday life!).

ARM borrows a bit from both RISC and CISC.