Introdução ao RISC-V

Conjuntos de Instruções do Processador - ISA

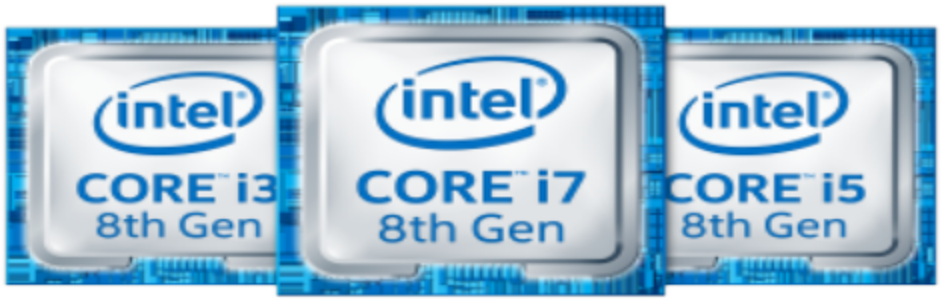
- *"The portion of the computer that is visible to the programmer or the compiler writer."* - Computer Architecture: A quantitative approach
- *"An instruction set architecture (ISA) is an abstract model of a computer. It is also referred to as architecture or computer arquiteture."* - Wikipedia
- *"A contract HW and SW designers agreed to obey"* - Minha definição de uma linha
- **"Um contrato em que os projetistas de hardware e software concordaram em obedecer"** - Minha definição de uma linha

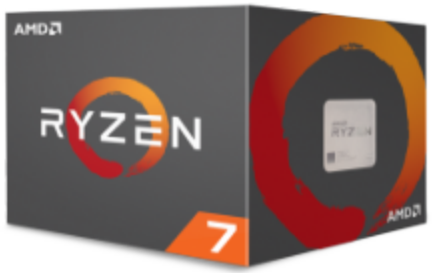
Arquitetura vs Microarquitetura

- Arquitetura é o modelo
  - x86, ARM, RISC-V, Power
- Microarquitetura é a implementação
  - Intel i7 geração 11, AMD Ryzen 3, ARM Cortex-A53, RISC-V RV32IMAC, PowerPC 970
- Conjunto de instruções pode ser visto como a borda
  - Pode facilitar ou dificultar a implementação em cada um dos lados

ISA é importante?

Instruction Set Architecture





Ambientes de Execução

Tipo	Sistema Operacional	Acesso aos Periféricos	Ambiente	Exemplo
Bare metal	Não	Direto	Memória	Arduino
Sistema Operacional	Sim	Indireto	Processo	Windows, Linux, iOS, Android
Hypervisor	Sim	Indireto	Máquina Virtual	VirtualBox, VMware, QEMU
Emulador	Sim	Indireto	Processo	MARS, QEMU

RISC-V: Características gerais

- Conjunto de instruções
  - Aberto
  - Modularizado
  - Virtualizável
- Licença de uso
  - Aberta
  - Não patenteada
- Espaço de endereçamento de **32**, 64 ou 128 bits
- Registradores de **32**, 64 ou 128 bits

Registradores do processador

- Registrador é um espaço de dados dentro do próprio processador que será utilizado para armazenar valores
- Nos seus primeiros programas, todas as variáveis estarão armazenadas em registradores
- O processador possui 32 registradores de uso geral
- No início da disciplina, utilizaremos apenas alguns deles
  - zero: registrador que tem sempre o valor zero
  - t0 - t6: 7 registradores para valores temporários
  - s0 - s11: 12 registradores para valores salvos

Primeiros formatos básicos das instruções

- mnemônico, rd, rs1, rs2

```
ADD s0, s1, s2      # soma o valor de s1 com o de s2 e armazena em s0
```

- mnemônico rs, rs1, imm

```
ADD s0, s1, 9      # soma o valor de s1 com 9 e armazena em s0
```

> *mnemônico* é o nome da instrução
>
> ⚠️ Cada instrução tem um e somente um formato, logo você não pode somar 3 valores utilizando apenas uma instrução como a primeira acima

Instruções Aritméticas

Instrução	Formato	Uso
ADD	R	ADD rd, rs1, rs2
ADD imediato	I	ADDI rd, rs1, imm
SUBtract	R	SUB rd, rs1, rs2
Load Upper Immediate	U	LUI rd, imm

Exemplos

- x = y + z

```
add t0, t1, t2   # onde t0 deve ter o valor de x, t1 de y e t2 de z
```

- x = x + y

```
add t0, t0, t1   # onde t0 deve ter o valor de x e t1 de y
```

- x = y + 7

```
addi t0, t1, 7   # onde t0 deve ter o valor de x, t1 de y
```

Mais exemplos

- x = y - z

```
sub t0, t1, t2   # onde t0 deve ter o valor de x, t1 de y e t2 de z
```

- x = x - y

```
sub t0, t0, t1   # onde t0 deve ter o valor de x e t1 de y
```

- x = y - 7

```
addi t0, t1, -7   # onde t0 deve ter o valor de x, t1 de y
```

> ⚠️ Não existe **subi**. Para subtrair um valor imediato, utilize **addi** com um valor negativo

Compondo um programa

- No simulador que utilizaremos nessa parte inicial da disciplina, o programa deve ser escrito em um arquivo de texto com extensão **.s** e deve começar declarando o label **main**. O programa deve terminar com a instrução **ret**.

```
main:
  addi t0, zero, 1          # t0 = 0 + 1
  addi t1, zero, 2          # t1 = 0 + 2
  add  t2, t1, t0           # t2 = t1 + t0
  ret
```

- **label:** indica a posição de um ponto no programa
- O que faz esse programa simples?

Como ler os valores da entrada e escrever na saída?

- Para ler um valor inteiro, é necessário utilizar as duas instruções abaixo. O resultado ficará no registrador **a0**:

```
addi t0, zero, 4    # escolhe a operação de leitura de inteiro (4)
ecall               # efetua a operação de leitura de inteiro
```

- Para escrever um valor inteiro, é necessário utilizar as duas instruções abaixo. O valor deve estar no registrador **a0**:

```
addi t0, zero, 1    # escolhe a operação de escrita de inteiro (1)
ecall               # efetua a operação de escrita de inteiro
```

Colocando tudo junto

- O programa abaixo lê dois valores inteiros e escreve a soma deles na saída:

```
main:
  addi t0, zero, 4    # escolhe a operação de leitura de inteiro (4)
  ecall               # efetua a operação de leitura de inteiro
  add s0, a0, zero    # guarda o valor retornado em a0 em s0
  addi t0, zero, 4    # escolhe a operação de leitura de inteiro (4)
  ecall               # efetua a operação de leitura de inteiro
  add  a0, s0, a0     # a0 = a0 + s0
  addi t0, zero, 1    # escolhe a operação de escrita de inteiro (1)
  ecall               # efetua a operação de escrita de inteiro
  ret
```

> Note que os registradores **t** do código original foram trocados por **s**
>
> ⚠️ O registrador **a0** é o único que pode ser utilizado para leitura e escrita de valores