

Aula 02 - Linguagem de Montagem - Assembly

▼ Tecnologias de Memória - Cai na prova

Linguagem de Montagem

- Arquitetura \Rightarrow ISA (Instruction Set Architecture) \Rightarrow Assembly (Ling. de Montagem) | Simuador MARS \rightarrow Simulador Nutella
- Usaremos o ISA da arquitetura MIPS (www.mips.com)
- Simulador \Rightarrow SPIM \rightarrow <http://spimsimulator.sourceforge.net> - `sudo apt install spim`
- Como usar o SPIM:
 1. Desenvolva o código no seu editor de preferência
 2. Rode no terminal
 - a. `spin-f codigo.spim`

Operações Aritméticas

\rightarrow Soma: add a, b, c

- O comando add é uma instrução (minemônico)
- a variável “a” vai ser o resultado, enquanto “b” e “c” serão os operandos.

$\Rightarrow a = b + c$

- As operações aritméticas sempre seguem esse padrão, primeiro vem o resultado e depois os operandos na ordem do comando, é um princípio de formato/design.

Princípio de Design 1: Simplicidade favorece a regularidade.

- \rightarrow Implementação simples.
- \rightarrow Melhora o desempenho.

Outras instruções aritméticas

→ sub - subtração

→ mul - multiplicação, essa instrução aritmética precisa de cuidado com o tanto de dígitos que o “a” consegue armazenar na base binária, o limite pode ser excedido facilmente.

Exemplo: $f = (g+h) - (i+j)$ | precisamos quebrar a operação aritmética de dois em dois usando o parênteses se necessário. O código exemplo primeiro precisa calcular o que está nos parênteses, depois calcular o resultado dos parênteses para fazer a subtração.

Quando usamos #, significa comentário em assembly, muito útil para entendermos o código assembly.

⇒ add t0, g, h # t0 = g+h

⇒ add t1, i, j # t1 = i+j

⇒ sub f, t0, t1 # t0 - t1

As “variáveis” que as instruções usam chamam-se registradores.

Registradores

- São pequenas unidades de memória que encontram-se dentro do processador
- A quantidade é limitada
- O tempo de acesso é quase imediato.
- O tamanho dos registradores é 32 bits / 4 bytes.
 - Por isso, nossa arquitetura é o MIPS 32 bits.

Um processador MIPS conta com 3 unidades de processamento:

1. Números inteiros: 32 registradores
2. Números reais: 32 registradores
3. Exceções - Não lideramos com isso nesta disciplina

Princípio de Design 2: menor é mais rápido.

Registradores de inteiros

- Como vários processos são rodados em 1 processador só? Ele consome em nanosegundos para realizar uma operação, então ele realiza bilhões de operações de 1 segundo, ai ele roda 500 mil para cada processo num sistema de escalonamento. Em 1 milisegundo ele processa 100 milhão de instruções, dividindo elas para cada processo.
- O processador não roda tudo ao mesmo tempo, ele divide o tempo dado e distribuí os processos no tempo que ele consegue executar tanta coisa, chamado de escalonamento de processo, utilizando a questão de alta ou baixa prioridade para quanto de instruções ele deve processar no escalonamento.
- Um processo sempre irá priorizar os "t"s e depois os "s".
- para \$s0-\$s7 em num temos 16-23.

Registadores de inteiros			
	Notação	Núm.	Descrição
	\$zero	0	Zero.
	\$at	1	Reservado assembler
	\$v0-\$v1	2-3	Res. de proced.
	\$a0-\$a3	4-7	Arg. de proced.
	\$t0-\$t7	8-15	Temporários
	\$s0-\$s7		Salvos

▼ **Porque cada registrador tem um número associado? - QUESTÃO PROVA**

- O computador e o processador entende 0 e 1, e os registradores devem estar associados aos números por isso, para que o computador entender.