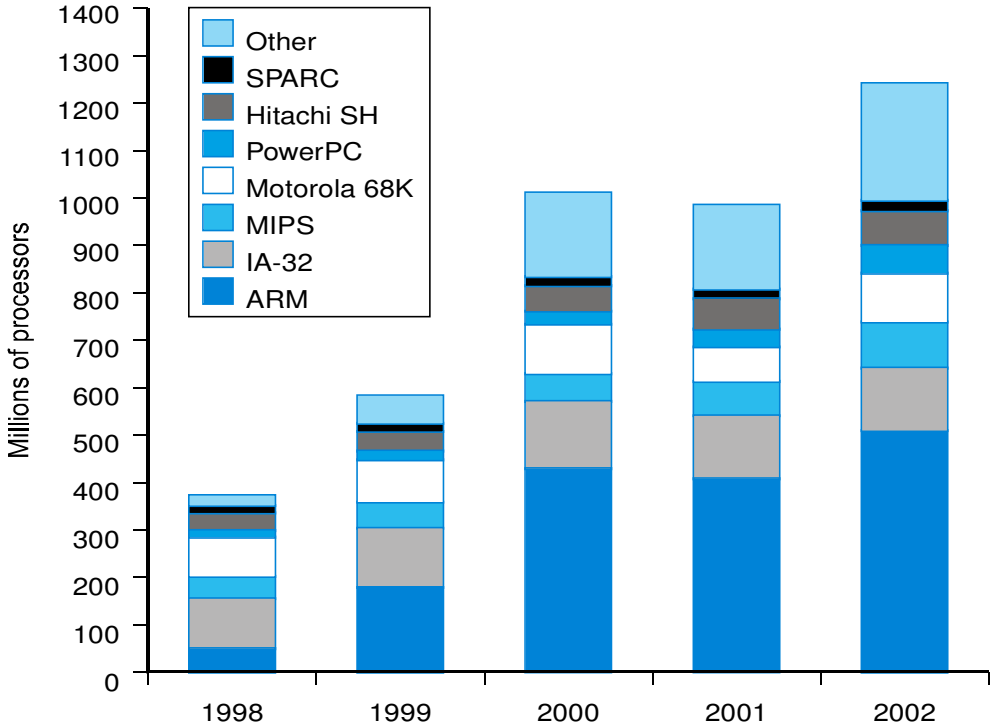
Arquitetura de Computadores I

Prof. Ricardo Santos
ricr.santos@gmail.com

(Cap 2)

Instruções

- “Comandos” utilizados para indicar ao hardware o que deve ser feito
- Utilizaremos neste curso o conjunto de instruções (ISA) do MIPS
  - Similar a outras arquiteturas desenvolvidas na década de 80
  - Quase 100 milhões de processadores MIPS manufaturados em 2002
  - Utilizado por NEC, Nintendo, Cisco, Silicon Graphics, Sony, …



Aritmética no Processador MIPS

- Operações com 3 operandos (1 destino e 2 fontes)
- Ordem dos operandos é fixa (destino primeiro)

Exemplo:

Código em C: `a = b + c`

'Código' MIPS : `add a, b, c`

"The natural number of operands for an operation like addition is three…requiring every instruction to have exactly three operands, no more and no less, conforms to the philosophy of keeping the hardware simple"

Aritmética no Processador MIPS

- Princípio de projeto: simplicidade favorece a regularidade
- Obviamente, algumas coisas ficam mais complexas...

Código em C:
```
a = b + c + d;
```

Código em MIPS:
```
add a, b, c
add a, a, d
```

- Operandos devem ser registradores, apenas 32 registradores estão disponíveis
- Cada registrador possui 32 bits

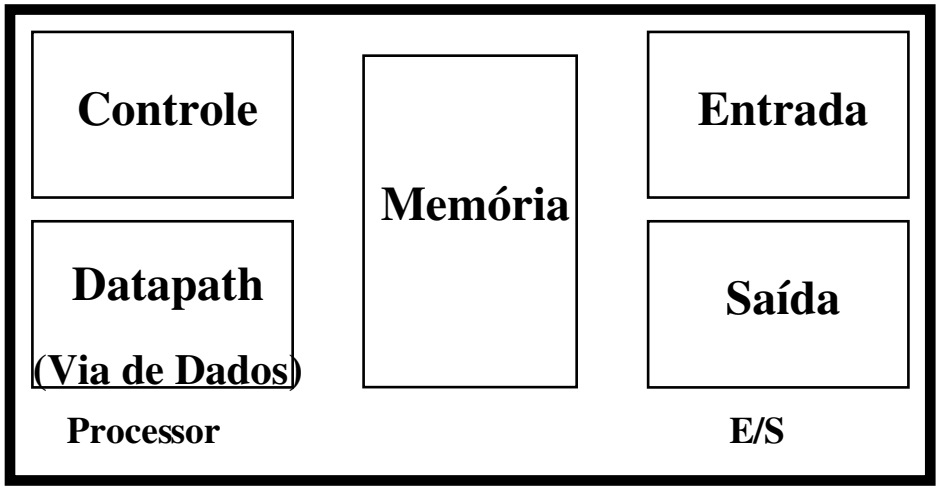
- Princípio de projeto: menor é mais rápido.

Alguma idéia do por quê disso?

Registradores vs Memória

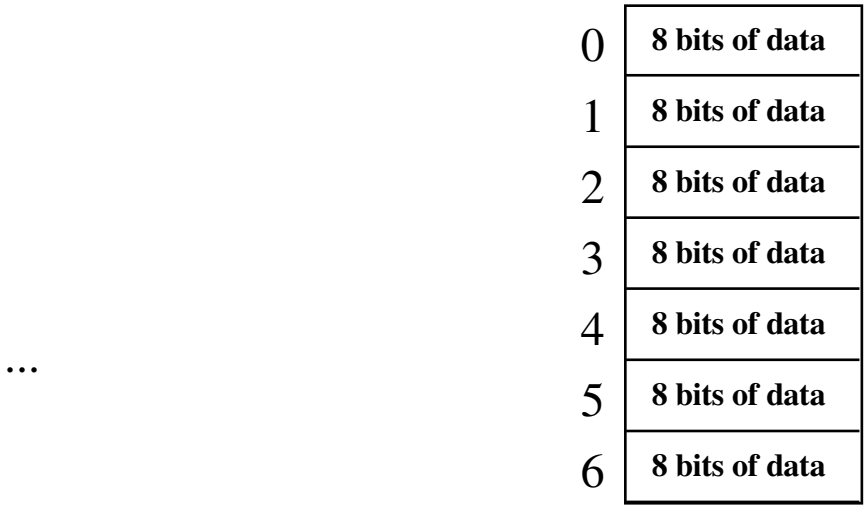
- Operandos de instruções aritméticas devem ser registradores

- O compilador associa as variáveis do programa a registradores
- Mas...e quanto aos programas que possuem mais que 32 variáveis?



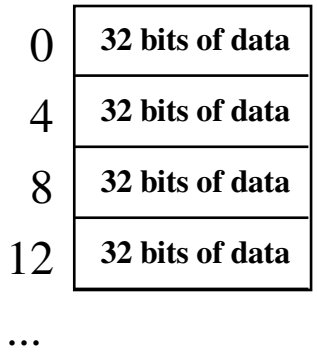
Organização da Memória

- Pode ser ‘entendida’ como um vetor unidimensional com um endereço
- Um endereço específico de memória corresponde a um índice desse vetor
- “Endereçamento de bytes" significa que os índices apontam para um byte da memória



Organização da Memória

- Muitos dados manipulados em programas são do tamanho de “palavras”
- No MIPS, uma palavra é 32 bits ou 4 bytes.



Registradores armazenam 32 bits de dados

- 2^{32} bytes com endereços de bytes de 0 to 2^{32}-1
- 2^{30} words com endereços de bytes 0, 4, 8, ... 2^{32}-4
- Palavras são alinhadas

Instruções

- Instruções de load (carregar) e store (armazenar)
- Exemplo:

Código C: `A[12] = h + A[8];`

Código MIPS:

```
lw $t0, 32($s3)
add $t0, $s2, $t0
sw $t0, 48($s3)
```

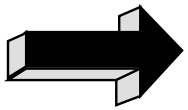
- Pode-se referir a registradores pelo nome (e.g., $s2, $t2) ao invés do número
- Na instrução de store, o destino aparece por último
- Lembre-se que operandos em instruções aritméticas são registradores, não é a memória

Está errado: `add 48($s3), $s2, 32($s3)`

Primeiro Exemplo

- Consegue descobrir quais variáveis correspondem a quais registradores?

```
swap(int v[], int k);
{ int temp;
    temp = v[k]
    v[k] = v[k+1];
    v[k+1] = temp;
}
```



```
swap:
    muli $2, $5, 4
    add $2, $4, $2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

Até Agora Aprendemos...

- MIPS
  — carrega-se palavras alinhadas em bytes
  — Instruções aritméticas trabalham apenas com registradores

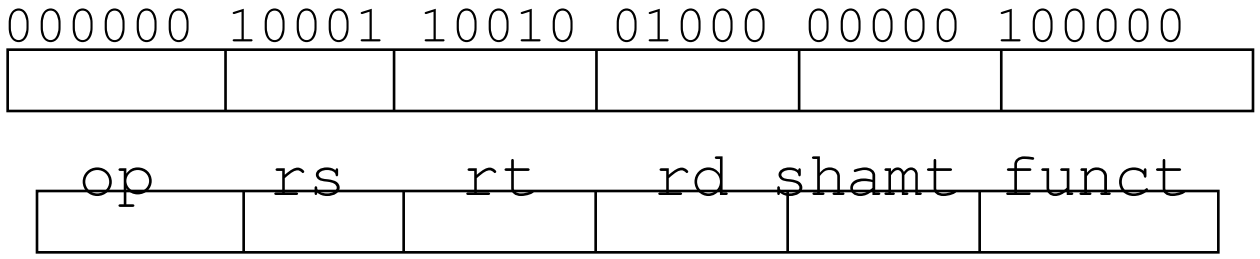
- Instrução — Significado

```
add $s1, $s2, $s3        $s1 = $s2 + $s3
sub $s1, $s2, $s3        $s1 = $s2 - $s3
lw $s1, 100($s2)         $s1 =
Memoria[$s2+100]
sw $s1, 100($s2)         Memoria[$s2+100] =
$s1
```

Instruções e Formatos

- Instruções, assim como registradores e palavras de dados, possuem 32 bits
  - Exemplo: `add $t1, $s1, $s2`
  - Registradores possuem números, `$t1=9, $s1=17, $s2=18`

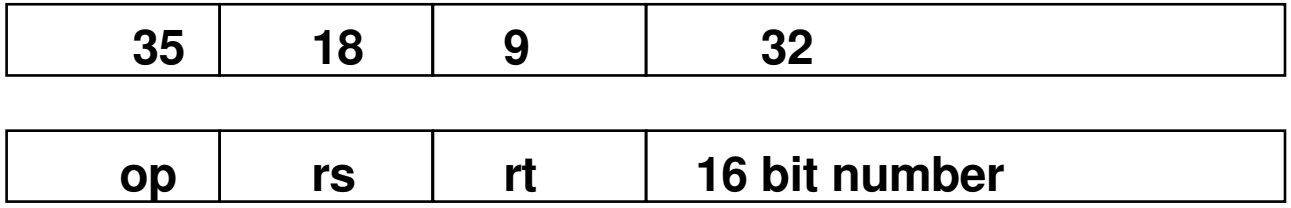
- Formato das instruções:



- *Qual o significado desses campos?*

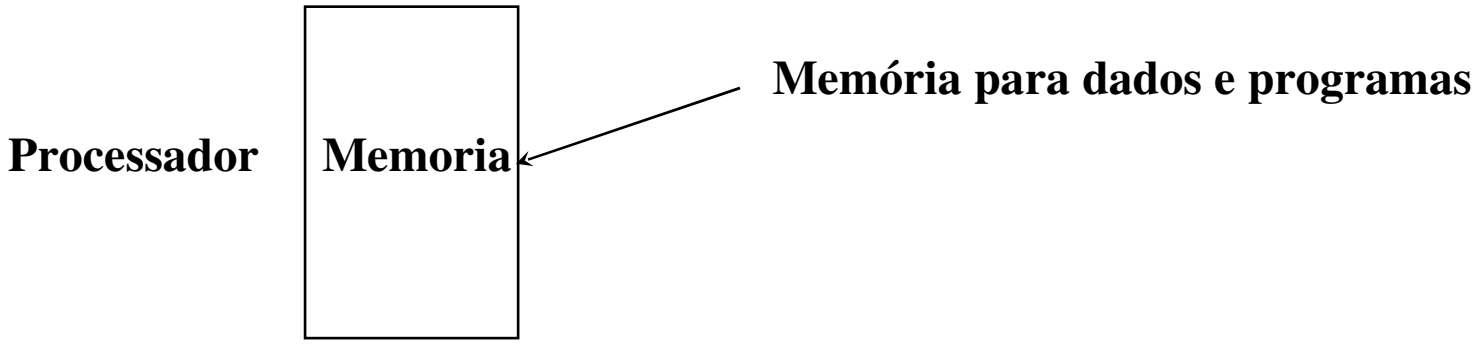
Instruções e Formatos

- Considere as instruções de *load* e *store*
- Deve-se introduzir um novo tipo de instrução
  - Tipo I, para instruções de transferência de dados
  - O formato para instruções aritméticas e o Tipo-R
- Exemplo: `lw $t0, 32($s2)`



Conceito de Programa Armazenado

- Arquitetura de Von Neumman
- Instruções são conjuntos de bits
- Programas são armazenados na memória
  — para serem lidos ou escritos assim como os dados



- Ciclos de Busca (Fetch) & Execução (Execute)
  - Instruções são buscadas e armazenadas em um registrador especial
  - Bits nesse registrador “controlam" as ações subsequentes
  - Buscar a “próxima” instrução e continuar

Instruções de Controle

- Instruções que “tomam decisões”
  - Alteram o fluxo de controle,
  - ou seja, mudam a próxima instrução a ser executada

- Instruções de desvio condicional do MIPS :

```
bne $t0, $t1, Label
beq $t0, $t1, Label
```

- Exemplo: `if (i==j) h = i + j;`

```
      bne $s0, $s1, Label
      add $s3, $s0, $s1
Label:    ....
```

Instruções de Controle

- Instruções de desvio incondicional do MIPS:

```
j  label
```

- Exemplo:

```
if (i!=j)          beq $s4, $s5, Lab1
    h=i+j;         add $s3, $s4, $s5
else               j Lab2
    h=i-j;         Lab1:sub $s3, $s4, $s5
                   Lab2:...
```

- A instrução de desvio incondicional é do Tipo-J
- *Tente construir um exemplo com o comando for!*

Até agora aprendemos:

Instrução	Significado
`add $s1,$s2,$s3`	`$s1 = $s2 + $s3`
`sub $s1,$s2,$s3`	`$s1 = $s2 - $s3`
`lw $s1,100($s2)`	`$s1 = Memoria[$s2+100]`
`sw $s1,100($s2)`	`Memoria[$s2+100] = $s1`
`bne $s4,$s5,L`	`Próxima instrução está em L se $s4≠ $s5`
`beq $s4,$s5,L`	`Próxima instrução está em L se $s4 = $s5`
`j Label`	`Próxima instrução está em Label`

- Formatos: