

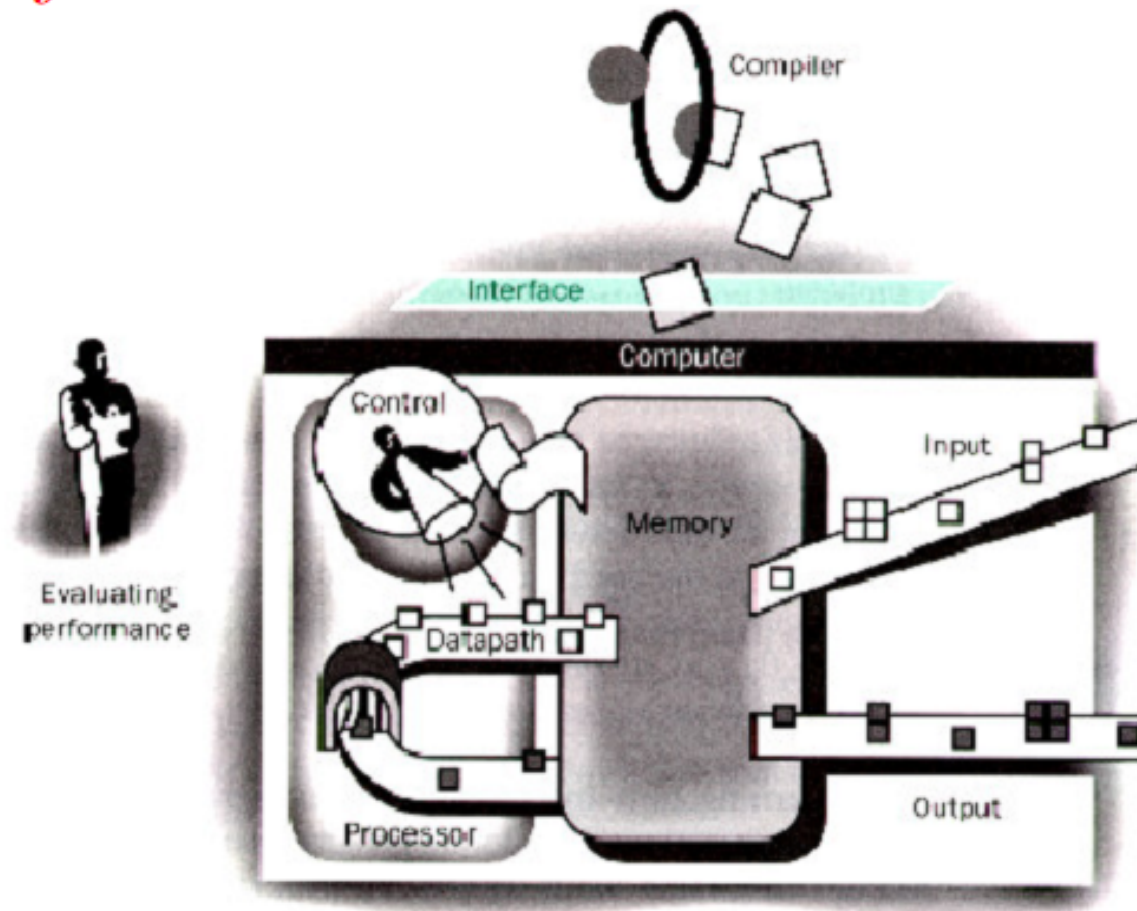


Infra-Estrutura de Hardware

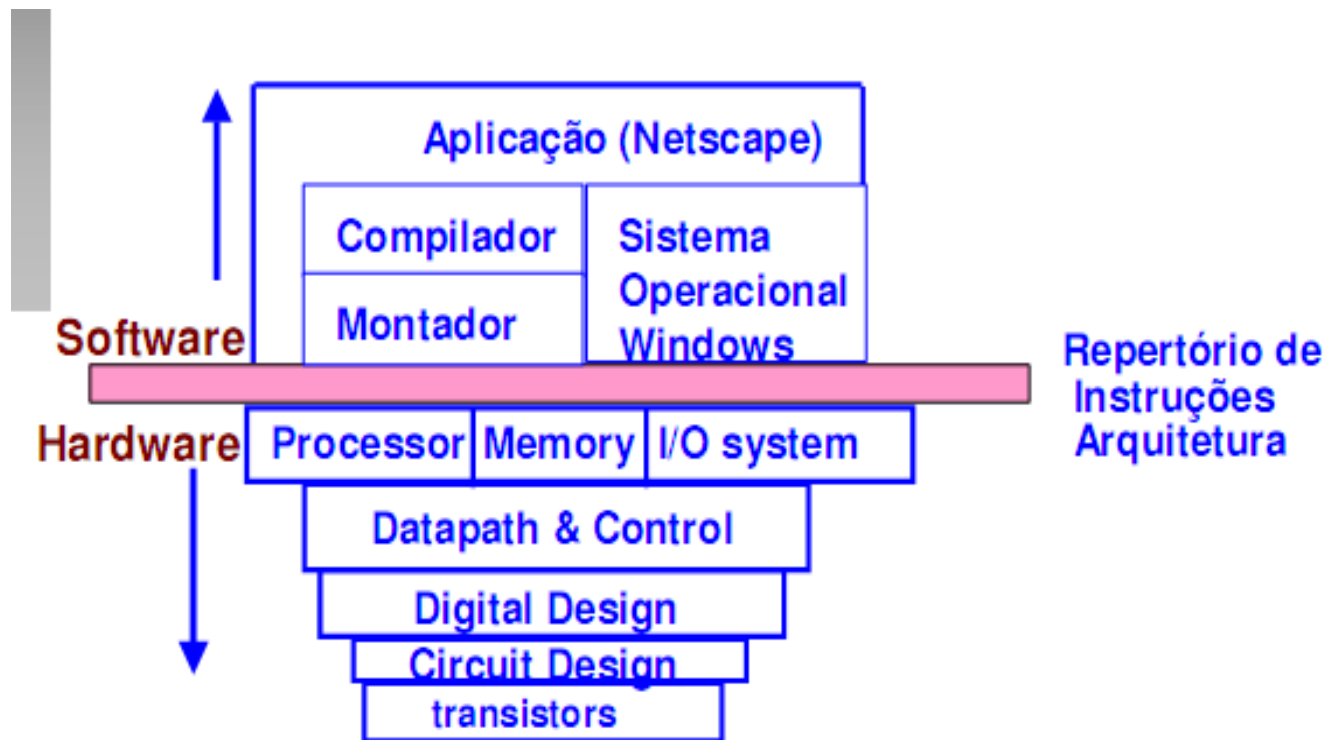
Conceitos Básicos de Arquitetura de Computadores

Universidade Federal Rural de Pernambuco
Professor: Abner Corrêa Barros
abnerbarros@gmail.com

Computador: Hardware + Software



Computador: Hardware + Software



Interface entre hw e sw: Repertório de Instruções

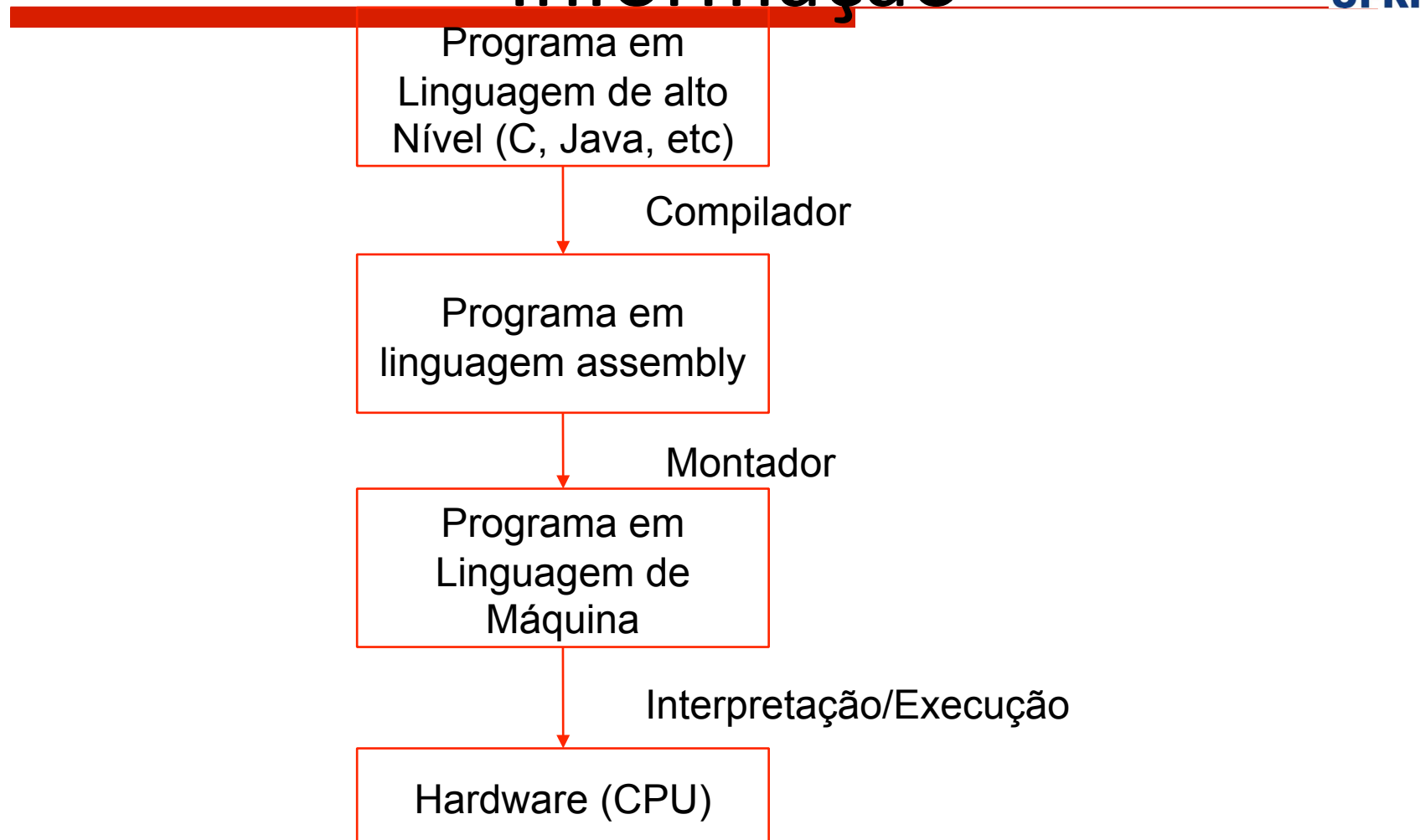


Think About...



- “To command a computer’s hardware, you must speak its language. The words of a computer’s language are called instructions, and its vocabulary is called an instruction set”
- Patterson

Representação da Informação



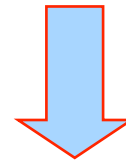
Representação da Informação



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

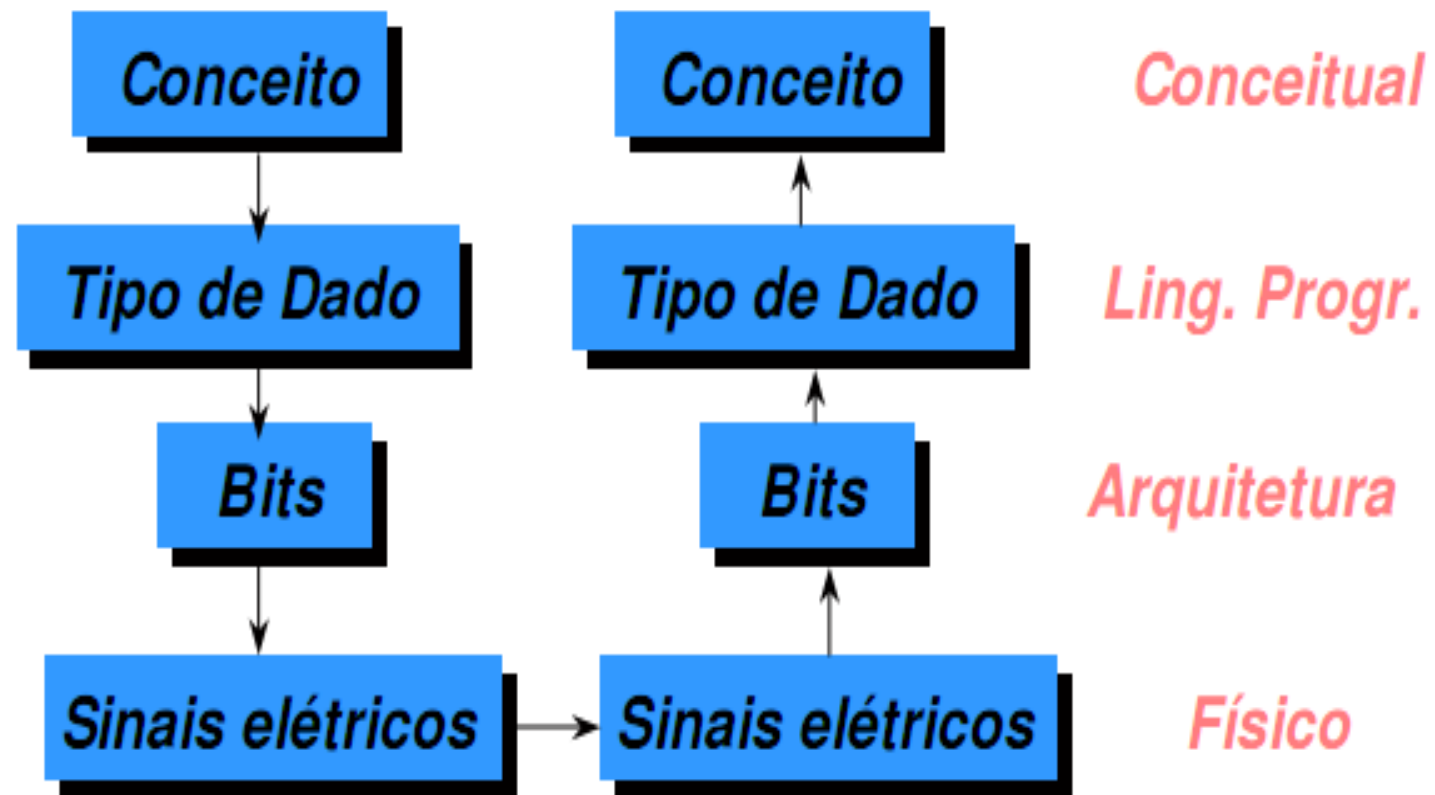


```
lw$t0, 0($2)  
lw$t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```



```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

Tipos de Dados



Tipos de Dados



- **Escalar**
 - Números
 - Inteiros
 - Ponto-Flutuante (real)
 - Caracteres
 - ASCII
 - EBCDIC
 - Dados lógicos
 - Variáveis booleanas (true/false)
- Obs: Podem ser armazenados tanto nos registradores do processador quanto na memória principal

Tipos de Dados



- Estruturados
 - Estático
 - Array
 - Record
 - Dinâmico
 - Listas
 - Árvores

Inteiros

Representação binária

– sinal-magnitude

0000000000001010
+ 10

1000000000001010
- 10

– complemento a 1

0000000000001010
+ 10

1111111111110101
- 10

– complemento a 2

0000000000001010
+ 10

1111111111110110
- 10

Dados lógicos



Representação

- *Uma palavra*

0000000000000000

Verdadeiro

0000000000000001

Falso

- *Um bit*

0000000000100000

Falso

Verdadeiro

Números no formato Ponto Flutuante



■ Representação

$$3,14 = 0,314 \times 10^1 = 3,14 \times 10^0$$

$$0,000001 = 0,10 \times 10^{-5} = 1,00 \times 10^{-6}$$

$$\rightarrow n = f \times 10^{e^-}$$

<i>sinal</i>	<i>expoente</i>	<i>mantissa (fração)</i>
--------------	-----------------	--------------------------

Caracteres




- Os caracteres são armazenados /manipulados através dos códigos numéricos que os representam, normalmente através do seu código ASCII (*American Standard Code for Information Interchange*) associado.
- Ex: Abner = 0x41,0x42,0x4e,0x45,0x52

Arrays



`char nome[] = "Luiz Inacio Lula da Silva";`



L 0x10	u 0x11	i 0x12	z 0x13
0x14	l 0x15	n 0x16	á 0x17
c 0x18	i 0x19	o 0x1A	0x1B
L 0x1C	u 0x1D	l 0x1E	a 0x1F

Obs: O conteúdo da variável fica armazenado na memória principal a partir do endereço atribuído à ela

Representação do Algoritmo



- Antes que possa ser compreendido e executado pelo processador, o algoritmo deve ser traduzido, compilado, para o conjunto de instruções da linguagem de montagem (*assembly*) do processador a que se destina

Representação do Algoritmo



- Em seguida, o algoritmo já traduzido para a linguagem assembly é convertido para o código de máquina do processador, transformando-se em sequência de palavras binárias que representam as instruções a serem executadas pelo processador.

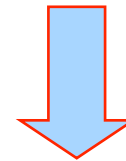
Compilação



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```



```
lw $t0, 0($2)  
lw $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

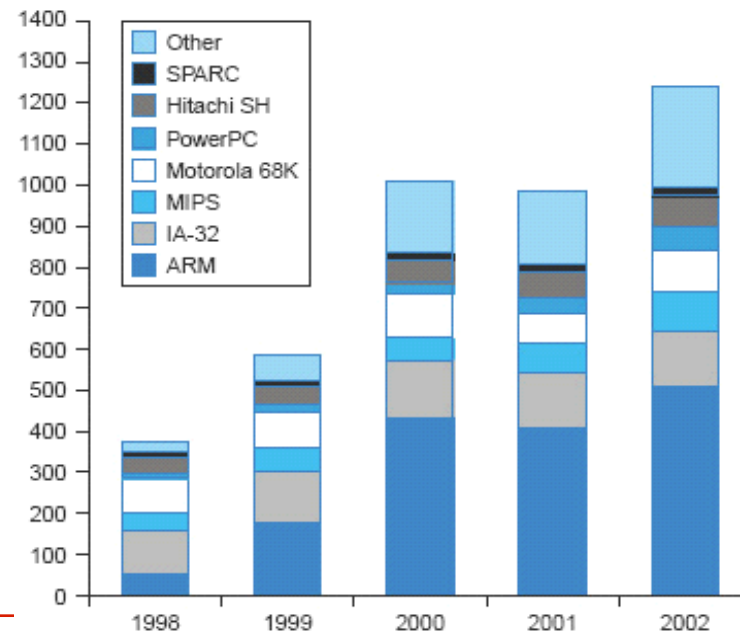


```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

Processador MIPS



- Vamos trabalhar com o processador MIPS, um dos processadores mais estudados e utilizados em todos os tempos.
- Quase 100 milhões de processadores MIPS fabricados em 2002
- Utilizado pela NEC, Nintendo, Cisco, Silicon Graphics, Sony...



Processador MIPS



- Tipos de instruções presentes na linguagem de montagem do MIPS:
 - Instruções Lógicas
 - Instruções Aritméticas
 - Instruções de Desvio
 - Instruções de Tomada de Decisão
 - Instruções de Transferencia de dados

Set de Instruções (MIPS)



Instrução	Descrição
nop	No operation
lw reg, end(reg_base)	reg. = mem (reg_base+end)
sw reg, end(reg_base)	Mem(reg_base+end) = reg
add regi, regj, regk	Regi. <- Regj. + Regk
sub regi, regj, regk	Regi. <- Regj. - Regk
and regi, regj, regk	Regi. <- Regj. and Regk
srl regd, regs, n	Desloca regs para direita logico n vezes e armazena em regd
sra regd, regs, n	Desloca regs para dir. aritm. N vezes e armazena em regd
sll regd, regs, n	Desloca regs para esquerda n vezes
ror regd, regs, n	Rotaciona regs para direita n vezes
rol regd, regs, n	Rotaciona regs para esquerda n vezes
beq regi, regj, desl	PC=PC+desl*4 se regi = regj
bne regi, regj, end	PC=PC+desl*4 se regi <> regj
slt regi, regj, regk	Regi = 1 se regj < regk senão regi=0
j end	Desvio para end
jr regd	Desvio para endereço em regd

Exemplo



- Instruções aritméticas no MIPS
 - $a = b + c$
add a, b, c
 - $a = (b+c)-(d+e)$
add t0, b, c # variáveis t0 e t1 são
add t1, d, e # variáveis auxiliares
sub a, t0, t1

Operandos em Hardware



- A fim de melhorar o desempenho e simplificar a implementação, o MIPS só executa operações lógicas e aritméticas com os operandos e o resultado das operações armazenados em registradores
 - O acesso aos registradores é sempre mais rápido que à memória.

Conjunto de registradores do MIPS



- O MIPS possui 32 registradores, assim distribuídos:

Nome	Número	Uso	Preservado em chamadas?
\$zero	0	Constante 0	n.d
\$v0-\$v1	2-3	Resultados e avaliações de expressões	Não
\$a0-\$a3	4-7	Argumentos	Sim
\$t0-\$t7	8-15	Temporários	Não
\$s0-\$s7	16-23	Salvos	Sim
\$t8-\$t9	24-25	Temporários	Não
\$gp	28	Ponteiro global	Sim
\$sp	29	Ponteiro para pilha	Sim
\$fp	30	Ponteiro para frame	Sim
\$ra	31	Endereço de retorno	Sim

Registradores especiais do MIPS



- \$zero (0): contém a constante “000...00”
 - Não pode ser modificado
- \$sp (29): Ponteiro da pilha (área especial de memória)
- \$ra (31): Endereço de retorno para a chamada de rotinas

Operandos na Memória



- Variáveis/constantes de tipos estruturados são sempre armazenadas na memória principal
- Variáveis/constantes de tipos escalares também podem estar armazenadas na memória principal
- Em ambos os casos, a fim de manipular com os dados armazenados na memória utilizam-se as instruções **load (lw e lb)** e **store (sw e sb)** para transportar os dados da memória para os registradores e vice-versa

Operandos na Memória



- Durante o processo de compilação do algoritmo, os nomes das variáveis e/ou constantes que ficam armazenadas na memória principal é substituída pela posição de memória a elas reservadas.
- Desta forma, uma variável de nome v1 que esteja armazenada na posição de memória 0x1234 terá todas as suas referências no código substituídas pelo valor 0x1234.

Operandos na Memória



- Exemplo:
 - Array no MIPS
 - Endereço da variável armazenado em Registrador
 - Elemento do array implícito na instrução
 - Ex: $A = \{0, 1, 2, 3, 4, \dots, 99\}$ // $A = \$s3$
 $g = h + A[8]$ // $g = \$s1$ e $h = \$s2$

 $lw \$t0, 8(\$s3)$ // $\$t0 = \text{mem}[\$s3+8]$
 $add \$s1, \$s2, \$t0$ // $\$s1 = \$s2 + \$t0$

Software de Apoio



- Como ambiente de desenvolvimento e estudo da arquitetura MIPS estaremos utilizando o aplicativo MipsIt, o qual permite compilar programas, simular o funcionamento do código obtido e inspecionar internamente o funcionamento do processador.

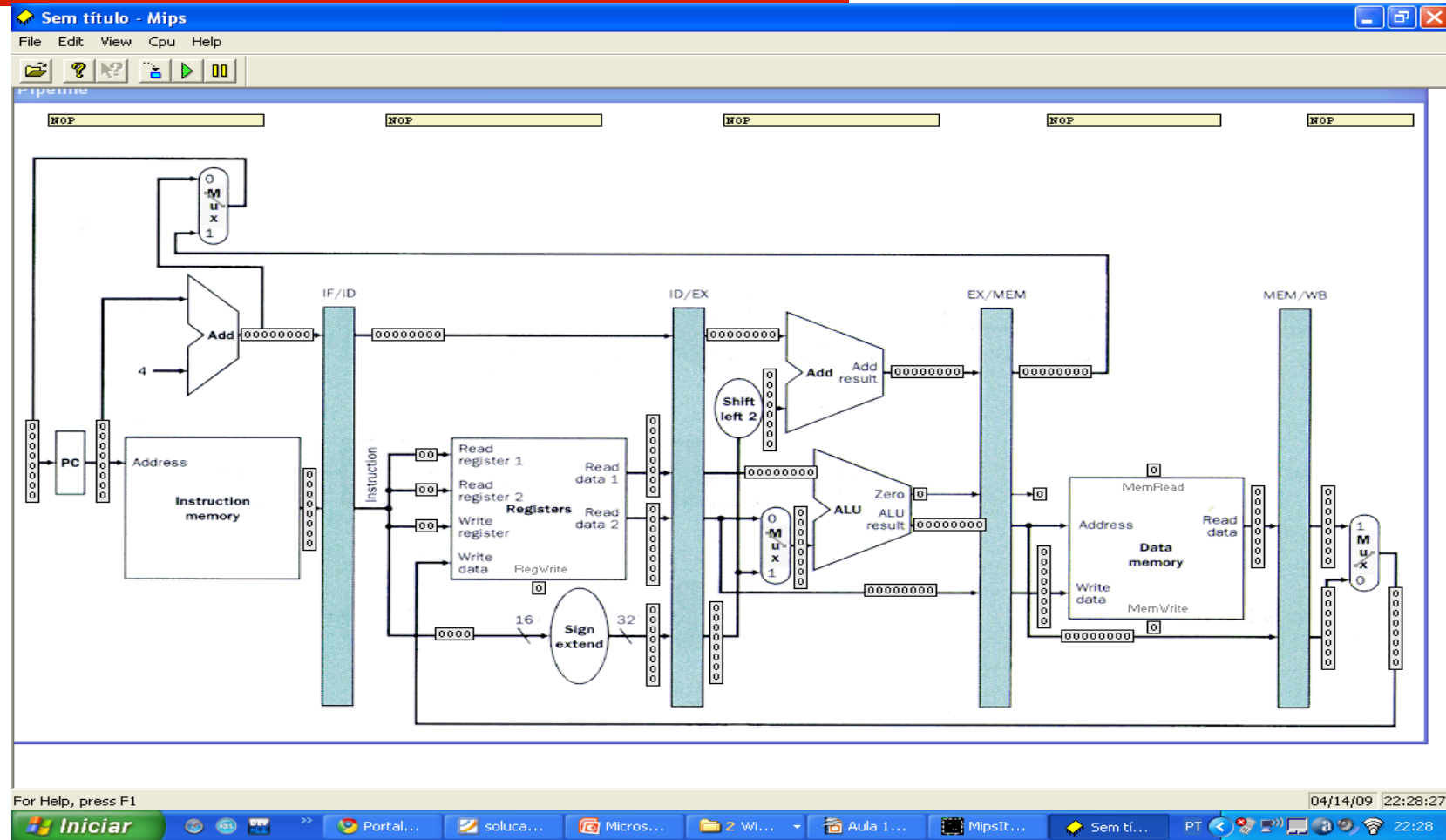
MipsIt



```
.data
a: .word 5
b: .word 10
c: .word 5
.text
.globl start
.ent start
start: lw $8, a
      lw $9, b
      lw $10, c
      add $11, $9, $8
      sub $11, $11, $10
      sw $11, a
.end start
```

abnerbarros@gmail.com

MipsIt



ExemploMIPS.srec - Mips

File Edit View Cpu Help

Memory

Address	Content	Label
8001FFC8	00 00 00 00	NOP
8001FFCC	00 00 00 00	NOP
8001FFD0	00 00 00 00	NOP
8001FFD4	00 00 00 00	NOP
8001FFD8	00 00 00 00	NOP
8001FFDC	00 00 00 00	NOP
8001FFE0	00 00 00 00	NOP
8001FFE4	00 00 00 00	NOP
8001FFE8	00 00 00 00	NOP
8001FFEC	00 00 00 00	NOP
8001FFF0	00 00 00 00	NOP
8001FFF4	00 00 00 00	NOP
8001FFF8	00 00 00 00	NOP
8001FFFC	00 00 00 00	NOP
80020000	3C 08 80 02	LUI \$08, 0x8002
80020004	8D 08 00 30	LW \$08, 0x30(\$08)
80020008	3C 09 80 02	LUI \$09, 0x8002
8002000C	8D 29 00 34	LW \$09, 0x34(\$09)
80020010	3C 0A 80 02	LUI \$10, 0x8002
80020014	8D 4A 00 38	LW \$10, 0x38(\$10)
80020018	01 28 58 20	ADD \$11, \$09, \$08
8002001C	01 6A 58 22	SUB \$11, \$11, \$10
80020020	3C 01 80 02	LUI \$01, 0x8002
80020024	AC 2B 00 30	SW \$11, 0x30(\$01)
80020028	00 00 00 00	NOP
8002002C	00 00 00 00	NOP
80020030	00 00 00 05	???

CPU

Registers

Register	Value
r0/zero	=00000000
r1/at	=00000000
r2/v0	=00000000
r3/v1	=00000000
r4/a0	=00000000
r5/a1	=00000000
r6/a2	=00000000
r7/a3	=00000000
r8/t0	=00000000
r9/t1	=00000000
r10/t2	=00000000
r11/t3	=00000000
r12/t4	=00000000
r13/t5	=00000000
r14/t6	=00000000
r15/t7	=00000000
r16/s0	=00000000
r17/s1	=00000000
r18/s2	=00000000
r19/s3	=00000000
r20/s4	=00000000
r21/s5	=00000000
r22/s6	=00000000
r23/s7	=00000000
r24/t8	=00000000
r25/t9	=00000000
r26/k0	=00000000
r27/k1	=00000000
r28/gp	=00000000
r29/sp	=800bc000
r30/fp	=00000000
r31/ra	=bfc00088
pc	=80020000
bad va	=00000000
mdhi	=00000000
mdlo	=00000000
conf	=00000000
status	=00400000
cause	=00000000
epc	=00000000

D-Cache

Address (00000000h)

31 8 3 2 0

00000000 0 0 0

V Tag Data

Max

Cache Hit Count Miss Count Hit Rate Cycle count

Console

For Help, press F1

04/14/09 22:27:16

PT 22:27