Comandos ou Instruções MIPS

Declaração:
Nome: storage_type value(s)
Exemplo1: .byte ‘a’, ‘b’
Exemplo2: .space 69

Adicionar	add $s1, $s2, $s3	$s1 = $s2 + $s3
Add imediato	addi $s1, $s2, C	$s1 = $s2 + C
Add unsigned imediato	addiu $s1, $2s, C	$s1 = $s2 + C
Add unsigned	addu $s1, $s2, $s3	$s1 = $s2 + $s3
And	and $s1, $s2, $s3	$s1 = $s2 and $s3
And imediato	andi $s1, $s2, C	$s1 = $s2 and C
Branch se igual	beq $s1, $s2, L	if($s1 == $s2) go to L
Branch se maior ou igual	bge $rs,$rt,Label	if($rs>=$rt) PC=Label
Branch se maior	bgt $rs,$rt,Label	if($rs>$rt) PC=Label
Branch se módulo maior	bgtu $rs,$rt,Label	if(|$rs\|>=\|$rt|) PC=Label
Branch se maior que zero	bgtz $rs,$rt,Label	if($rs>0) PC=Label
Branch se menor ou igual	ble $rs,$rt,Label	if($rs<=$rt) PC=Label
Branch se menor	blt $rs,$rt,Label	if($rs<$rt) PC=Label
Branch se diferente	bne $s1, $s2, L	if($s1 != $s2) go to L
Break	break	usado para debugar
Clear	clear $rt	$rt=0
Quociente da divisão	div $d, $s, $t	$d = $s / $t
Jump	j 2500	Desvia para 10000
Jump and link	jal C	usado para chamar sub-rotina
Jump register	jr $s1	Desvia para $s1
Load Address	la $at, LabelAddr	$at = Label Address
Load double word	ld $t,C($s)	$t = Mem[C+$s]
Load Immediate	li $at, IMMED[31:0]	$at = 32 bit Immediate value
Load Upper Immediate	lui $s1, 100	$s1 = 100 * 2
Load word	lw $s1, 100($s2)	$s1 = Mem[$s2+100]
Move	move $rt,$rs	$rt = $rs
Mult(primeiros 32 bits)	mul $d, $s, $t	$d = $s * $t
Nop	nop	interpretado como sll $0, $0, 0
Nor	nor $s1, $s2, $s3	$s1 = $s2 nor $s3
Ou	or $s1, $s2, $s3	$s1 = $s2 or $s3
Ou imediato	ori $s1, $s2, C	$s1 = $s2 or C
Resto da divisão	rem $d, $s, $t	$d = $s % St

Shift left logical	sll $t, $s, C	$t = $s << C
Set on less than	slt $s1, $s2, $s3	if($s2 < $s3) $s=1; else $s1=0
Set less than immediate	slti $s1, $s2, 100	if($s2 < 100) $s=1; else $s1=0
Shift right logical	srl $t, $s, C	$t = $s >> C
Subtrair	sub $s1, $s2, $s3	$s1 = $s2 – $s3
Subtrair unsigned	subu $s1, $s2, $s3	$s1 = $s2 – $s3
Store word	sw $s1, 100($s2)	Mem[$s2+100] = $s1
Syscall	syscall	chama sistema operacional
Xor	xor $s1, $s2, $s3	$s1 = $s2 xor $s3

Tabela de Registradores do MIPS, e suas funções:

Registrador Número	Nome Alternativo	Descrição
0	zero	Registrador sempre zerado
1	$at	Temporário para o montador
2-3	$v0 – $v1	Para atribuições de expressões e resultados de funções
4-7	$a0 – $a3	Argumentos para subrotinas
8-15	$t0 – $t7	Temporários
16-23	$s0 – $s7	Valores
24-25	$t8 – $t9	Temporários
26-27	$k0 – $k1	Reservado para uso de interruptores
28	$gp	Ponteiro global
29	$sp	Ponteiro para pilha
30	$s8/$fp	Ponteiro frame
31	$ra	Endereço de Retorno

versão: 0.71

Instruções editadas por Andrei Costa.
Contato: andrei529@msn.com

Entrada/saída de dados por chamadas ao sistema operativo

O SPIM disponibiliza um conjunto de serviços semelhantes aos do SO através da utilização da instrução ***syscall*** (**chamada ao sistema**)**:**

- código do serviço → $v0
- argumentos → $a0-$a3 ($f12 com reais)
- valor devolvido → $v0 ($f0 com reais)

Serviço	Código	Argumentos	Resultado
print_int	**1**	**$a0** = inteiro	
print_float	**2**	**$f12** = float	
print_double	**3**	**$f12** = double	
print_string	**4**	**$a0** = string	
read_int	**5**		inteiro em **$v0**
read_float	**6**		float em **$f0**
read_double	**7**		double em **$f0**
read_string	**8**	**$a0** = *buffer*, **$a1** = tamanho máximo	
sbrk	**9**	**$a0** = quantidade (n)	endereço em **$v0**
exit	**10**		

Descrição dos serviços

- **print_int, print_float, print_double** → escreve no ecrã o valor presente no registo adequado ao tipo de serviço
- **print_string** → escreve no ecrã a cadeia de caracteres terminada pelo caracter **'\0'**
- **read_int, print_float, print_double** → lê um número do teclado e guarda-o no registo adequado ao tipo de serviço
- **read_string** → tem a funcionalidade da função **fgets()** do **C** (lê até se atingir $a1-1 caracteres ou se encontrar o caracter '\n')
- **sbrk** → devolve o apontador para um bloco de memória com **n** bytes livres
- **exit** → suspende a execução do programa

Exemplo: imprimir a cadeia de caracteres “A resposta e’ 5”

```
.data
str:
.asciiz "A resposta e’ "    # string terminada por ‘\0’
.text
     li $v0, 4       # identificação do serviço print_str
     la $a0, str     # endereço da cadeia de caracteres a
                     #   imprimir
     syscall         # invocação do serviço print_str
     li $v0, 1       # identificação do serviço para print_int
     li $a0, 5       # valor a imprimir
     syscall         # invocação do serviço print_int
```

Entrada/saída de dados por sondagem directa no SPIM

- O SPIM simula um periférico: um terminal mapeado em memória. Isto impede que se use simultaneamente entradas/saídas mapeadas em memória
- Para se poder usar entradas/saídas mapeadas em memória, o simulador deverá ser reiniciado com a opção ***-mapped_io***
- O terminal consiste em duas unidades independentes: um emissor (que escreve no ecrã) e um receptor (que lê do teclado)
- O terminal é controlado pelos programas através de 4 registos mapeados em memória