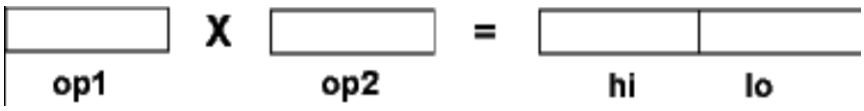
MIPS – Lab. Arq. Computadores
Novas instruções mult, div mfhi e mflo.

Consulte as transparências para as instruções:
Jump (j)
Branch if equal (beq)
Branch if not equal (bne)
Set on less then (slt)

Unidade de Multiplicação no MIPS

A unidade multiplicadora do Mips possui dois registradores de 32-bit denominados **hi** e **lo**. Não são registradores de propósito geral. Quando dois registradores com operandos de 32-bit são multiplicados, **hi** e **lo** possuem o resultado de 64 bits.



Bits 32 até 63 estão em **hi** e bits 0 até 31 em **lo**.

mult e multu

```
mult s,t        # hilo <-- $s * $t (operandos em com. de dois)
multu s,t       # hilo <-- $s * $t (operandos unsigned )
```

A multiplicação inteira nunca causa um trap.

Perg.: Dois inteiros pequenos são multiplicados, onde está o resultado?
Resp: Se o resultado for inferior a 32 bits, em lo, hi possuirá zeros.

Bits Significantes

Os bits significantes em um num. Positivo ou unsigned são todos os bits à direita do bit com valor 1 mais à esquerda do número:

0000 0000 0100 0011 0101 0110 1101 1110
O número possui 23 bits significantes.

Os bits significantes em um num. Negativo, são todos os bits à direita do bit com valor 0 mais à esquerda do número:

1111 1111 1011 1100 1010 1001 0010 0010
O número possui 23 bits significantes.

Para garantir que o produto não possuirá mais do que 32 bits, a soma dos bits

significantes dos números deverá ser menor ou igual a 32.

Perg.: Aproximadamente quantos bits significantes você espera do produto:
01001010 × 00010101
Resp: Aproximadamente 12 bits.

mfhi e mflo

São as instruções utilizadas para mover os resultados da multiplicação para registradores de uso geral.

```
mfhi d          # d <-- hi. Mover de Hi
mflo d          # d <-- lo. Mover de Lo
```

Perg: Deve-se mover os resultados de lo e hi antes de uma multiplicação seguinte?
Resp: Sim.

Perg.: Como computar 5 × X - 74 ?

Resp.:

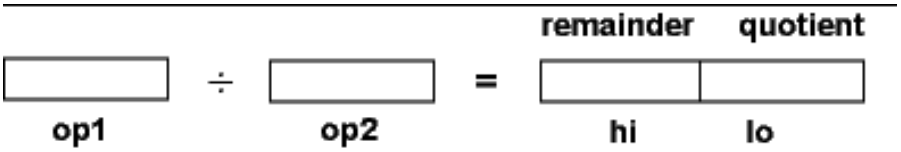
```
## novoMult.asm
##
## Programa para calcular 5 × x - 74
##
## $8 x
## $9 resultado
.text
.globl main
main:
ori $8, $0, 12              # x em $8
ori $9, $0, 5               # 5 em $9
mult $9, $8                 # lo <-- 5x
mflo $9                             # $9 = 5x
addi $9, $9,-74             # $9 = 5x - 74
## End of file
```

Perg.: O que significa o "u" nas seguintes instruções:
addu _____
multu_____

Resp:
addu, não acontece um trap no overflow.
multu, os operandos são unsigned.

div e divu

De maneira análoga, temos em hi e lo o resultado e o quociente, respectiamente.




```
div s,t         # lo <-- s div t
                # hi <-- s mod t
                # operandos em comp. de dois
divu s,t        # lo <-- s div t
                # hi <-- s mod t
                # operandos em comp. De dois
```

Perg.: Calcular (y + x) / (y - x)

Resp.:

```
## exemplodiv.asm
##
## Programa para calcular (y + x) / (y - x)
##
## $8 x
## $9 y
## $10 x/y
## $11 x%y
.text
.globl main
main:
ori $8, $0, 8           # x em $8
ori $9, $0, 36          # y em $9
add $10, $9, $8         # $10 <-- (y+x)
sub $11, $9, $8         # $11 <-- (y-x)
div $10, $11            # hilo <-- (y+x)/(y-x)
mflo $10                        # $10 <-- quociente
mfhi $11                        # $11 <-- resultado
## End of file
```

Perg.: (36+8) / (36-8) =
Resp: (36+8) / (36-8) = 1 R 16, or 0x1 R 0x10

PROGRAMAS

// programa 17

1) $y = \begin{cases} x^4 + x^3 - 2x^2 & \text{se x for par} \\ x^5 - x^3 + 1 & \text{se x for impar} \end{cases}$

Os valores de x devem ser lidos da primeira posição livre da memória e o valor de y deverá ser escrito na segunda posição livre.

// programa 18

2) $y = \begin{cases} x^3 + 1 & \text{se } x > 0 \\ x^4 - 1 & \text{se } x <= 0 \end{cases}$

Os valores de x devem ser lidos da primeira posição livre da memória e o valor de y deverá ser escrito na segunda posição livre.

// programa 19

Escreva um programa que avalie a expressão: (x*y)/z.

Use x = 1600000 (=0x186A00), y = 80000 (=0x13880), e z = 400000 (=0x61A80). Inicializar os registradores com os valores acima.

// programa 20

Escreva um programa que gere um vetor de inteiros até 100. Seu programa deverá armazenar na memória os números pares separados dos ímpares. Armazene primeiro os pares e logo a seguir os ímpares.
Mostre a tabela de porcentagens das instruções utilizadas.

// programa 21

Para a expressão a seguir, escreva um programa que calcule o valor de k:

$k = x^y$

O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.

Pseudo Instruções

São instruções que permitem a escrita do código de maneira mais rápida, entretanto são formadas através da execução de uma seqüência de instruções nativas.

No exemplos a seguir temos diversas pseudo instruções (move, li, la e lw), use o MARS para verificar quais são as instruções nativas utilizadas em cada uma das pseudo instruções do programa.

```
.text
.globl main
main:

#.....................................................

move $t1,$t0            #move de t0 para t1, traduzida por addu

#.....................................................

li $t0, 0x12345678    # carrega o registrador com um imediato de até 32 bits,
li $t1, -12345678      # usa $at assembler temporary  traduzida como lui e ori,
                       # pode-se usar um numero negativo

#.....................................................

la $t0, varx            # carrega o registrador com o endereço simbólico,
la $t1, vary           # usa os registradores at, e instr. lui e ori
                       # para determinarmos o endereço.
#.....................................................

lw $t0, varx           # carrega o registrador com o valor do endereço simbólico,
lw $t1, vary           # usa os registradores at, e instr. lui e ori
                       # para determinarmos o endereço.
                       # ATENCÃO: não confundir com o lw $reg, imediato($reg)


.data
varx: .word 3
vary: .word 5
```

Responda:
Qual registrador o MIPS usa como um temporário do assembler?

Serviços do Sistema Operacional (Handlers) no MIPS

Service	Code in $v0	Arguments	Returned Value
print integer	1	$a0 == integer	
print float	2	$f12 == float	
print double	3	($f12, $f13) == double	
print string	4	$a0 == address of string	
read integer	5		$v0 == integer
read float	6		$f0 == float
read double	7		($f0, $f1) == double
read string	8	$a0 == buffer address $a1 == buffer length	
allocate memory	9	$a0 == number of bytes	$v0 == address
exit	10		

Exemplo 1:

```
.text
.globl main
main:

sll $0,$0, 0


li $v0,4       # código 4 == print string
la $a0,string  # $a0 == endereço da string
syscall        # Solicita o serviço ao SO


li $v0,5       # código 5 == read integer
syscall        # Solicita serviço ao SO
               # Lê a linha em caracteres ascii
               # Converte para um inteiro de 32-bit
               # $v0 <-- recebe o inteiro


li $v0,1       # code 1 == print integer
lw $a0,numero  # $a0 = o que está no endereço numero
```

```
syscall         # Solicita serviço ao SO
                # Converte inteiro de 32-bit em caracteres
                # Mostra caracteres no monitor

li $v0,10       # código 10 == exit
syscall         # Retorna controle ao SO



.data
string: .asciiz "Alo MIPS!\n Enfim sei programar !!!\n"
numero: .word 100
# end of file
```

Exemplo 2:

```
## soma.asm
## programa para adicionar dois inteiros armazenados na
memória
##

la $t0,val2          # carrega um endereço de 32-bit em $t0
lw $t1,0($t0)        # carrega primeiro valor, 2
lw $t2,val3          # carrega Segundo valor, 3
sll $0,$0,0          # load delay slot
add $t3,$t1,$t2      # calcula a soma
sw $t3, val4          # escreve na memória



.data
val1: .word 1
val2: .word 2
val3: .word 3
val4: .word 4
```

Programas / Relatório:

Programa 22
Escreva um programa que escreva na tela: “Hello world !”

Programa 23
Escrever um programa que solicite que ao usuário digite 3 números, em seguida determinar o maior número e o menor número. O programa deve mostrar na tela: “O maior número = ...” e “O menor = ... “

Programa 24
Escrever um programa que leia a quantidade de elementos de um vetor (Solicitar ao usuário que digite esse número).
Em seguida você deverá criar um vetor com a seguinte série: 1, 3, 5, 7, 9, ...
Após montado o vetor, seu programa deverá percorrer esse vetor, calcular a soma de todos os elementos e mostrar na tela essa soma com o texto: “A soma dos elementos = ...”

Programa 25

Você deverá elaborar um programa que solicite uma temperatura em graus Fahrenheit, em seguida o programa deverá converter a temperatura lida para graus Celsius.
Obs.: O programa deverá mostrar apenas 1 casa decimal (use truncamento).

A fórmula de conversão é: $°C = 5 / 9 . (°F − 32)$

Exemplo:
Tela: Qual a temp. em Fahrenheit?
Usuário : 40
Tela: A temp. em Celsius e:
Tela: 4.4