

TARJETA DE REFERENCIA DE MIPS/MARS

PRINCIPALES INSTRUCCIONES (INCLUYENDO ALGUNAS PSEUDO-INSTRUCCIONES)

TIPO	NOMBRE	INSTRUCCIÓN	OPERACIÓN	NOTA
Aritméticas	Suma	add rd,rs,rt	$R[rd] = R[rs] + R[rt]$	(1)
	Suma sin signo	addu rd,rs,rt	$R[rd] = R[rs] + R[rt]$	(2)
	Suma con inm.	addi rd,rs,inm	$R[rt] = R[rs] + \text{InmSignExt}$	(1)(2)
	Suma sin signo con inm.	addiu rd,rs,inm	$R[rt] = R[rs] + \text{InmSignExt}$	(2)
	Resta	sub rd,rs,rt	$R[rd] = R[rs] - R[rt]$	(1)
	Resta sin signo	subu rd,rs,rt	$R[rd] = R[rs] - R[rt]$	
	Divide	div rs,rt	$Lo = R[rs] / R[rt]; Hi = R[rs] \% R[rt]$	
	Divide y obtiene cociente	div rd,rs,rt	$Lo = R[rs] / R[rt]; Hi = R[rs] \% R[rt]; R[rd] = Lo$	(1)
	Divide y obtiene resto	rem rd,rs,rt	$Lo = R[rs] / R[rt]; Hi = R[rs] \% R[rt]; R[rd] = Hi$	(1)
	Divide sin signo	divu rs,rt	$Lo = R[rs] / R[rt]; Hi = R[rs] \% R[rt]$	(6)
	Multiplica	mult rs,rt	$\{Hi, Lo\} = R[rs] * R[rt]$	
	Multiplica y obtiene result.	mul rd,rs,rt	$\{Hi, Lo\} = R[rs] * R[rt]; R[rd] = Lo$	
	Multiplica sin signo	multu rs,rt	$\{Hi, Lo\} = R[rs] * R[rt]$	(6)
Obtener valor en (Hi,Lo)	Mueve desde Hi	mfhi rd	$R[rd] = Hi$	
	Mueve desde Lo	mflo rd	$R[rd] = Lo$	
Lógicas bit a bit	Y lógico	and rd,rs,rt	$R[rd] = R[rs] \& R[rt]$	
	Y lógico con inm.	andi rt,rs,inm	$R[rt] = R[rs] \& \text{InmSignExt}$	(2)
	O lógico negado	nor rd,rs,rt	$R[rd] = \sim(R[rs] R[rt])$	
	O lógico	or rd,rs,rt	$R[rd] = R[rs] R[rt]$	
	O lógico con inm.	ori rt,rs,inm	$R[rt] = R[rs] \text{InmSignExt}$	(2)
	O exclusivo	xor rd,rs,rt	$R[rd] = R[rs] \wedge R[rt]$	
	O exclusivo con inm.	xori rt,rs,inm	$R[rt] = R[rs] \wedge \text{InmSignExt}$	(2)
Negación	Negación	not rd,rs	$R[rd] = \sim(R[rs])$	
Desp. de bits	Desp. lóg. a la izquierda	sll rd,rs,valor	$R[rd] = R[rs] \ll \text{valor}$	
	Desp. lóg. a la derecha	srl rd,rs,valor	$R[rd] = R[rs] \gg \text{valor}$	
	Desp. arit. a la derecha	sra rd,rs,valor	$R[rd] = R[rs] \gg \text{valor}$	
	Desp. lóg. a la izq. var.	sllv rd,rs,rt	$R[rd] = R[rs] \ll R[rt]$	
	Desp. lóg. a la der. var.	srlv rd,rs,rt	$R[rd] = R[rs] \gg R[rt]$	
	Desp. arit. a la der. var.	srav rd,rs,rt	$R[rd] = R[rs] \gg R[rt]$	
Comparación	1 si menor que	slt rd,rs,rt	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	
	1 si menor que sin signo	sltu rd,rs,rt	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	(6)
	1 si menor que inm.	slti rt,rs,inm	$R[rt] = (R[rs] < \text{InmSignExt}) ? 1 : 0$	(2)
	1 si menor que inm. sin sig.	sltiu rt,rs,inm	$R[rt] = (R[rs] < \text{InmSignExt}) ? 1 : 0$	(2)(6)
Salto condicional	Salta si igual	beq rs,rt*,dir	if ($R[rs] == R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si no igual	bne rs,rt*,dir	if ($R[rs] != R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si > 0	bgtz rs,dir	if ($R[rs] > 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si >= 0	bgez rs,dir	if ($R[rs] \geq 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si < 0	bltz rs,dir	if ($R[rs] < 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si <= 0	blez rs,dir	if ($R[rs] \leq 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si igual a 0	beqz rs,dir	if ($R[rs] == 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si distinto de 0	bnez rs,dir	if ($R[rs] != 0$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si menor que	blt rs,rt*,dir	if ($R[rs] < R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si mayor que	bgt rs,rt*,dir	if ($R[rs] > R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si menor o igual que	ble rs,rt*,dir	if ($R[rs] \leq R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	Salta si mayor o igual que	bge rs,rt*,dir	if ($R[rs] \geq R[rt]$) $PC = PC + 4 + \text{DirSalRel}$	(4)
	(*) Se puede sustituir rt por una constante			
Salto incondicional	Salta	j dirección	$PC = \text{DirSalAbs}$	(5)
	Salta y enlaza	jal dirección	$R[31] = PC + 4; PC = \text{DirSalAbs}$	(5)
	Salta según registro	jr rs	$PC = R[rs]$	
	Salta y enlazar según reg.	jalr rs	$R[31] = PC + 4; PC = R[rs]$	
Copia de reg.	Mover (copiar)	move rd,rs	$R[rd] = R[rs]$	
Asignar constante a registro	Asigna inm. en mitad sup.	lui rt,inm	$R[rt] = \{\text{inm}, 16'b0\}$	
	Asigna inm.	li rd,inm	$R[rd] = \text{inmediato}$	
	Asigna dirección	la rd,dirección	$R[rd] = \text{dirección}$	
Carga de dato de memoria	Carga byte	lb rt,inm(rs)	$R[rt] = \{24\{\text{signo}\}, M[R[rs] + \text{InmSignExt}](7:0)\}$	(2)
	Carga byte sin signo	lbu rt,inm(rs)	$R[rt] = \{24'b0, M[R[rs] + \text{InmSignExt}](7:0)\}$	(2)
	Carga media palabra	lh rt,inm(rs)	$R[rt] = \{16\{\text{signo}\}, M[R[rs] + \text{InmSignExt}](15:0)\}$	(2)
	Carga media pal. sin signo	lhu rt,inm(rs)	$R[rt] = \{16'b0, M[R[rs] + \text{InmSignExt}](15:0)\}$	(2)
	Carga palabra	lw rt,inm(rs)	$R[rt] = M[R[rs] + \text{InmSignExt}]$	(2)
Almacena. de dato en memoria	Almacena byte	sb rt,inm(rs)	$M[R[rs] + \text{InmSignExt}](7:0) = R[rt](7:0)$	(2)
	Almacena media palabra	sh rt,inm(rs)	$M[R[rs] + \text{InmSignExt}](15:0) = R[rt](15:0)$	(2)
	Almacena palabra	sw rt,inm(rs)	$M[R[rs] + \text{InmSignExt}] = R[rt]$	(2)

- (1) Puede provocar excepción por desbordamiento
- (2) $\text{InmSignExt} = \{16\{\text{inmediato}[15]\}, \text{inmediato}\}$
- (3) $\text{InmCeroExt} = \{16'b0, \text{inmediato}\}$

- (4) $\text{DirSalRel} = \{14\{\text{inmediato}[15]\}, \text{inmediato}, 2'b0\}$
- (5) $\text{DirSalAbs} = \{\text{PC}[31:28], \text{dirección}, 2'b0\}$
- (6) Los operandos se consideran números sin signo

REGISTROS ENTEROS

NOM.	NÚM.	USO	¿SALVAR?
\$zero	0	Valor constante 0	—
\$at	1	Temporal del ensamblador	No
\$v0-\$v1	2-3	Resultados de funciones y evaluación de expresiones	No
\$a0-\$a3	4-7	Argumentos de funciones	No
\$t0-\$t7	8-15	Temporales	No
\$s0-\$s7	16-23	Temporales preservados	Sí
\$t8-\$t9	24-25	Temporales	No
\$k0-\$k1	26-27	Reservados para el SO	No
\$gp	28	Puntero global	Sí
\$sp	29	Puntero de pila	Sí
\$fp	30	Puntero de marco de pila	Sí
\$ra	31	Dirección de retorno	Sí

DIR. DE MEMORIA EN CARGA/ALMACEN.

EXPRESIÓN	DIRECCIÓN DE MEMORIA
inm(rs)	$\text{dir} = \text{R}[\text{rs}] + \text{InmSignExt} (2)$
etiqueta	$\text{dir} = \text{etiqueta}$
etiqueta+inm	$\text{dir} = \text{etiqueta} + \text{InmSignExt} (2)$
etiqueta(rs)	$\text{dir} = \text{R}[\text{rs}] + \text{etiqueta}$
etiqueta+inm(rs)	$\text{dir} = \text{R}[\text{rs}] + \text{etiqueta} + \text{InmSigExt} (2)$

DIRECTIVAS DEL ENSAMBLADOR

.data <i>[dir]*</i>	Lo siguiente se almacena en el segmento de datos
.text <i>[dir]*</i>	Lo siguiente se almacena en el segmento de código * Comenzando en la dirección <i>[dir]</i> si se especifica
.ascii <i>cad</i>	Almacena la cadena <i>cad</i> en memoria, sin terminarla con NULL (' $\backslash 0$ ')
.asciiz <i>cad</i>	Almacena la cadena <i>cad</i> en memoria, añadiendo NULL (' $\backslash 0$ ') al final
.byte b_0, \dots, b_{n-1}	Almacena los n valores en bytes consecutivos de memoria
.half h_0, \dots, h_{n-1}	Almacena alineados n valores de 16 bits en medias palabras consecutivas de mem.
.word w_0, \dots, w_{n-1}	Almacena alineados n valores de 32 bits en palabras consecutivas de memoria
.space n	Reserva n bytes de espacio en el segmento actual y los inicializa a 0
.globl <i>eti</i>	Declara que la etiqueta <i>eti</i> es global y puede ser referenciada desde otros ficheros
.align n	Alinea lo siguiente a una dirección múltiplo de 2^n

ESTRUCTURA DE UN PROGRAMA

.data
<Reserva de espacio para datos>
<Importante respetar alineamiento>
.text
<Declaración de funciones o procedimientos>
<Importante respetar convención uso registros>
.globl main
main:
<Función donde comienza el programa>

LLAMADAS AL SISTEMA (SELECCIONADAS)

SERVICIO	DESCRIPCIÓN	\$v0	ARGUMENTOS	RESULTADO
print_int	Imprime entero	1	Entero \$a0	
print_string	Imprime cadena	4	Cadena \$a0	
read_int	Lee entero	5		Entero (en \$v0)
read_string	Lee cadena	8	Buffer \$a0, tamaño \$a1	
sbrk	Reserva memoria montón	9	Bytes a reservar \$a	Dirección (en \$v0)
exit	Termina sin código de salida	10		
print_char	Imprime carácter	11	Carácter \$a0	
read_char	Lee carácter	12		Carácter (en \$v0)
exit2	Termina con código de salida	17	Valor de retorno \$a0	
time	Obtiene la hora del sistema (milisegundos desde 1-1-1970)	30		32 bits inferiores (en \$a0) 32 bits superiores (en \$a1)
sleep	Pausa la ejecución	32	Miliseg. de pausa \$a0	
clear_screen	Limpia (borra) la pantalla	39		
random_int	Obtiene n ^o aleatorio	41	Id. del generador \$a0	Entero aleatorio (en \$a0)
random_int_range	Obtiene n ^o aleatorio en rango	42	Id. del generador \$a0, límite superior \$a1	Entero aleatorio (en \$a0)