

Registres

Nom	Numéro	Utilisation
\$zero	0	Constante 0
\$at	1	Réservé
\$v0	2	Résultat de fonction
\$v1	3	Résultat de fonction
\$a0	4	Paramètre 1
\$a1	5	Paramètre 2
\$a2	6	Paramètre 3
\$a3	7	Paramètre 4
\$t0	8	Temporaire
\$t1	9	Temporaire
\$t2	10	Temporaire
\$t3	11	Temporaire
\$t4	12	Temporaire
\$t5	13	Temporaire
\$t6	14	Temporaire
\$t7	15	Temporaire
\$s0	16	Temporaire (sauvé)
\$s1	17	Temporaire (sauvé)
\$s2	18	Temporaire (sauvé)
\$s3	19	Temporaire (sauvé)
\$s4	20	Temporaire (sauvé)
\$s5	21	Temporaire (sauvé)
\$s6	22	Temporaire (sauvé)
\$s7	23	Temporaire (sauvé)
\$t8	24	Temporaire
\$t9	25	Temporaire
\$k0	26	Réservé
\$k1	27	Réservé
\$gp	28	Pointeur global
\$sp	29	Pointeur de pile
\$fp/\$s8	30	Pointeur de cadre
\$ra	31	Adresse de retour

Branchements / Sauts

b étiquette	Branchement inconditionnel
beq rs, imm/rt, étiquette	Branch if Equal
beqz rs, étiquette	Branch if Equal to Zero
bge rs, imm/rt, étiquette	Branch if Greater or Equal
bgeu rs, imm/rt, étiquette	Branch if Greater or Equal Unsigned
bgez rs, étiquette	Branch if Greater or Equal to Zero
bgt rs, imm/rt, étiquette	Branch if Greater Than
bgtu rs, imm/rt, étiquette	Branch if Greater Than Unsigned
bgtz rs, étiquette	Branch if Greater Than Zero
ble rs, imm/rt, étiquette	Branch if Lower or Equal
bleu rs, imm/rt, étiquette	Branch if Lower or Equal Unsigned
blez rs, étiquette	Branch if Lower or Equal to Zero
blt rs, imm/rt, étiquette	Branch if Lower Than
bltu rs, imm/rt, étiquette	Branch if Lower Than Unsigned
bltz rs, étiquette	Branch if Lower Than Zero
bne rs, imm/rt, étiquette	Branch if Not Equal
bnez rs, étiquette	Branch if Not Equal to Zero
j étiquette	Jump
jal étiquette	Jump And Link
jr rs	Jump from Register
jalr rs	Jump And Link from Register

Stockage

sb rs, étiquette/imm(rt)	MEM[étiquette]:8 = rs / MEM[rt + imm]:8 = rs
sh rs, étiquette/imm(rt)	MEM[étiquette]:16 = rs / MEM[rt + imm]:16 = rs
sw rs, étiquette/imm(rt)	MEM[étiquette]:32 = rs / MEM[rt + imm]:32 = rs
move rd, rs	rd = rs

Chargement

la rd, étiquette	rd = @étiquette
lb rd, étiquette/imm(rs)	rd = MEM[étiquette]:8 / rd = MEM[rs + imm]:8
lh rd, étiquette/imm(rs)	rd = MEM[étiquette]:16 / rd = MEM[rs + imm]:16
li rd, imm	rd = SignExt(imm)
lui rd, imm	rd = (imm << 16) & 0xffff0000
lw rd, étiquette/imm(rs)	rd = MEM[étiquette]:32 / rd = MEM[rs + imm]:32

Directives d'assemblage

.ascii chaîne	Stocke chaîne en mémoire
.asciiz chaîne	Stocke chaîne suivie de '\0'
.byte b1, ..., bn	Stocke n octets consécutivement
.data	Stockage dans le segment de données
.globl étiquette	étiquette est un symbole global
.half h1, ..., hn	stocke n demi-mots (16 bits)
.space n	Réserve la place pour n octets
.text	Stockage dans le segment de code
.word w1, ..., wn	Stocke n mots (32 bits)

Arithmétique entière

abs rd, rs	Valeur absolue (rd = rs)
add rd, rs, imm/rt	Addition signée (rd = rs + [imm/rt])
addu rd, rs, rt	Addition non signée (rd = rs + rt)
div rd, rs, rt	Division signée (rd = rs / rt)
divu rd, rs, rt	Division non signée (rd = rs / rt)
mul rd, rs, rt	Multiplication (rd = rs * rt)
mulo rd, rs, rt	Multiplication avec erreur sur dépassement (rd = rs * rt)
mulou rd, rs, rt	Multiplication non signée avec erreur sur dépassement (rd = rs * rt)
neg rd, rs	Négation (rd = -rs)
rem rd, rs, rt	Reste de la division signée (rd = rs % rt)
remu rd, rs, rt	Reste de la division non signée (rd = rs % rt)
sub rd, rs, imm/rt	Soustraction signée (rd = rs - [imm/rt])
subu rd, rs, rt	Soustraction non signée (rd = rs - rt)

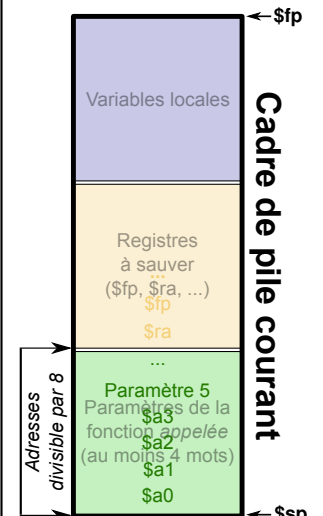
Instructions logiques

and rd, rs, imm/rt	ET bit à bit (rd = rs & [imm/rt])
not rd, rs	NON bit à bit (rd = ~rs)
or rd, rs, imm/rt	OU bit à bit (rd = rs [imm/rt])
sll rd, rs, imm	Shift Left Logical (rd = rs << imm)
sllv rd, rs, rt	Shift Left Logical Variable (rd = rs << rt)
sra rd, rs, imm	Shift Right Arithmetic (rd = rs >> imm)
srav rd, rs, rt	Shift Right Arithmetic Variable (rd = rs >> rt)
srl rd, rs, imm	Shift Right Logical (rd = rs >> imm)
srlv rd, rs, rt	Shift Right Logical Variable (rd = rs >> rt)
xor rd, rs, imm/rt	OU Exclusif (rd = rs ^ [imm/rt])

Appels système

Service	Appel (\$v0)	Entrée	Sortie
print_int	1	\$a0: entier à afficher	
print_string	4	\$a0: adresse chaîne	
read_int	5		\$v0
read_string	8	\$a0: adresse de stockage \$a1: taille maximum	
malloc	9	\$a0: taille à allouer	
exit	10		
print_char	11	\$a0: caractère (octet faible)	
read_char	12		\$v0
print_hex	34	\$0a: entier à afficher	
print_uint	36	\$0a: entier à afficher	
rand_seed	40	\$a0: valeur de l'initialiseur	
random	41	\$a0: n'importe quel entier	\$a0

Pile d'appels



Hello World!

```

; Déclaration des données
.data
hello_str: .asciiz "Hello World\n"

.text
.globl __start

; Programme principal
__start:
    la $a0, hello_str
    li $v0, 4
    syscall

; exit
li $v0, 10
syscall
    
```