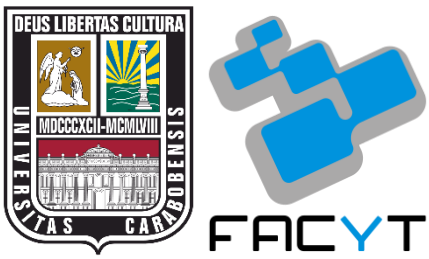

Hoja De Referencia para Mips32
Edael Duque C.I. 30800814

Instrucciones Fundamentales de Mips32

Para registros:

- $zero y $0 (Register 0): Contiene valor 0 permanente (No puede escribirse sobre él). Tiene usos como inicialización de registros, también se usa en operaciones donde el 0 es requerido.
- $at (Assembler Temporary): Se reserva para el ensamblador y está restringido para el programador, traduce las pseudo instrucciones a través de una conversión.
- $v0 y $v1 (Value): Su uso reside en el retorno de valores de funciones. $v0 como registro principal para el valor que retorna y como respaldo para un segundo valor se usa $v1 en casos donde el problema lo amerite.
- De $a0 hasta $a3 (Argument): Se utilizan como los primeros cuatro argumentos de una función.
- De $t0 hasta $t9 (Temporary): Su utilidad consiste en almacenar valores que no necesitan ser almacenados durante las llamadas a una función, por eso su nombre de temporales.
- De $s0 hasta $s7 (Saved): Registros de guardado con función de almacenar valores que es menester guardar para el problema en cuestión durante las llamadas de funciones.
- $k0 y $k1 (Kernel): Se reservan para el sistema operativo.
- $gp (Global Pointer): Es un puntero para los datos globales. Para acceder con mayor facilidad a variables globales y estáticas.
- $sp (Stack Pointer): Puntero para la pila, se encarga de apuntar al tope de la pila con el objeto de gestionar su marco de llamadas a funciones.
- $fp (Frame Pointer): Puntero de marco. Apunta a una posición fija dentro del marco de la pila actual. No se usa con tanta frecuencia.
- $ra (Return Address): Es una dirección de retorno. Se encarga de almacenar una dirección de la instrucción seguido de una llamada a función en condición de jal.
-

Para las operaciones aritméticas:

- add: para sumar dos registros y guardar el valor en un tercer registro.
- sub (subtract): para restar dos registros y guardarlo en un tercer registro.
-

Para la transferencia de datos:

- lw (load word): Mueve una palabra de una dirección a un registro.
- sw (store word): Mueve una palabra de un registro a una dirección.
- lh (load half): Carga la mitad de una palabra de memoria a un registro.
- sh (store half): Carga la mitad de una palabra de un registro a memoria.
- lb (load byte): Carga un byte de memoria a un registro.
- sb store byte): Carga un byte de un registro a memoria.
- lui (load upper inmediate): carga un valor constante de 16 bits en la parte superior de un registro.

Para operaciones lógicas

- and: Trabaja bit por bit en tres registros, utiliza una compuerta AND.
- or: Trabaja bit por bit en tres registros, utiliza una compuerta OR.
- not: trabaja bit por bit en un registro, utiliza una compuerta NOT.
- ori (or inmediate): variante de or que maneja bit por bit un registro y una constante, utiliza también una compuerta OR.
- sll (shift left logical): es un desplazamiento lógico hacia la izquierda de bit multiplicado a una constante.
- srl (shift right logical): es un desplazamiento lógico hacia la derecha de bits multiplicado por una constante.

Para el salto condicional:

- beq (branch on equal): Comprueba si dos registros son iguales y salta si lo son.
- bne (branch on not equal): Comprueba si dos registros no son iguales y salta si no lo son.
- slt (set on less than): Compara si el primer registro es menor que el segundo, y pone 1 en un registro destino si es verdad, 0 si es falso.
- slti (set on less than immediate): Compara si un registro es menor que una constante, y pone 1 en un registro destino si es verdad, 0 si es falso.

Para el salto incondicional:

- j (jump): Salta a una dirección de destino especificada.
- jr (jump register): Salta a la dirección contenida en un registro (comúnmente usado para retornar de un procedimiento).
- jal (jump and link): Salta a una dirección de destino y guarda la dirección de retorno (la siguiente instrucción) en el registro $ra (usado para llamadas a procedimientos).