Tema-2-Lenguaje-del-computador-M...

CodeWolf

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Accede al documento original

Tema 2: Lenguaje del computador MIPS

➢ Arquitectura con Acumulador:

Las primeras computadoras tenían esta arquitectura implementa ; teniendo un único registro para instrucciones aritméticas. La principal desventaja es que todas las variables del programa deben de estar ubicadas en memoria. Por lo cual ↑ Acceso a memoria ↓ Velocidad en la ejecución.

➢ Procesador MIPS

- ➔ Diseñado en la Universidad de Stanford.
- ➔ Procesador que utiliza una arquitectura de registros de propósito general .(CPU de 32 registros de 32 bits).
- ➔ Utiliza arquitectura RISC:
  - ◆ Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.
  - ◆ Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.
- ➔ El banco de registros está formado por 32 registros de 32 bits:
  - ◆ Dos puertos de lectura.
  - ◆ Un puerto de Escritura
- ➔ Proporcionando una mayor flexibilidad a la hora de realizar las instrucciones.
  - ◆ Implicará plantearse nuevos formatos de instrucción.

★ Operandos en registros.

La notación que se utiliza es: (instrucción) registro donde se guarda, registro 1 , registro _2

➢ Instrucciones MIPS:

- ➔ **add** a , b , c # b + c => a
- ➔ **sub** a, b, c # b - c => a
- ➔ **lw** $ t0 , 4 ($s1) # Carga lo de la derecha en la izquierda
- ➔ **sw** $t0 , 4 ($s1) # Guarda lo de la derecha en la izquierda
- ➔ **beq** registro 1, registro 2 , L1 # Salta a la sentencia L1 si el valor del reg1 es igual a reg2
- ➔ **bne** registro 1, registro 2 , L1 # Salta a la sentencia L1 si el valor del reg1 no es igual a reg2
- ➔ **slt** reg resul, reg1, reg2 # reg_resul= 1 si reg1 < reg2 en caso contrario reg_resul =0;
- ➔ **addi** reg guar , reg1 , constantes # reg guar = reg1 + constantes
- ➔ **slti** reg_resul , reg1 ,constantes # reg_resul = 1 si reg1 < constantes
- ➔ **lui**
- ➔ **j** dirección # Salta a la dirección . Usa el Tipo-J (Formato)
- ★ ***Desplazamientos lógicos***
- ➔ **sll** reg1 ,reg2, 4 # Transfiere el valor de reg2 desplazado 4 bits a la izquierda a reg1
- ➔ **srl** reg1 ,reg2, 4 # Transfiere el valor de reg2 desplazado 4 bits a la derecha a reg1

★ ***Operaciones AND lógica y OR lógica***

➔ **and** reg_result ,reg1 ,reg2 # reg_result = reg1 & reg2

➔ **or** reg_result ,reg1 ,reg2 # reg_result = reg1 | reg2

➔ **andi**

➔ **ori**

➔ **lb** $t0, 4($s2) # Carga un byte de memoria (derecha) en la izquierda

➔ **sb** $t0 , 4($s2) # Guarda los 8 bits menos significativos del registro fuente

➔ **jal** Dirección del Procedimiento # Sala la dirección donde continúa el programa guardando la dirección de retorno en el registro $ra

➔ **jr** $reg1 # Salta a la dirección contenida en $reg1

MIPS debe incluir instrucciones que transfieren datos entre memoria y registros . En este caso se guarda la dirección " base " .

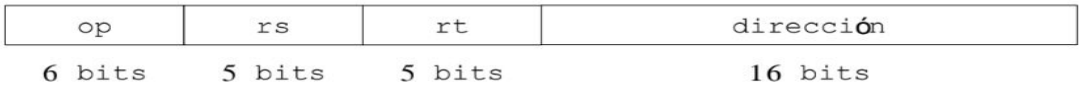
★ La memoria principal utilizada por MIPS es de 32 bits por palabra y utiliza 32 bits para ser direccionada.

- El elemento mínimo direccionable es el byte. Como cada palabra contiene 4 bytes, los dos últimos bits de dirección se encargará de seleccionar el byte dentro de la palabra.
- Por tanto, las direcciones consecutivas difieren de 4 .

Las instrucciones MIPS son de 32 bits y están divididas en los siguientes campos: **(TIPO R)**

Donde:

- **op:** Código de operación.
- **rs:** Primer registro de operando fuerte
- **rt;** Segundo registro de operando fuerte.
- **rd:** Registro operando destino.
- **shamt :** Tamaño del desplazamiento
- **funct :** Función, Selecciona la variable específica de la operación op.



Para las instrucciones de lw y sw se define otro formato : **(TIPO I)**

★ ***Formato para los saltos incondicionales :***

Utilizado para la instrucciones j , jal y jr. **(TIPO J)**