Imagen que contiene dibujo, señal, reloj

Descripción generada automáticamente

**ING. en Computación**

**Seminario de Solución de Problemas de Arquitectura de Computadoras "D12"**

**Profesor: Jorge Ernesto López Arce Delgado**



***PROYECTO FINAL: PROGRAMACIÓN EN VERILOG Y LENGUAJE ENSAMBLADOR***

**Equipo:**

**Estrada Huerta Félix Eduardo**

**Ortega Morales Juan José**

**González Ramírez Carlos Arturo**

**01 - Junio - 2021.   
Guadalajara, Jalisco**

**CARACTERISTICAS GENERALES DEL PROCESADOR MIPS DE 32 BITS**

MIPS (Microprocessor without Interlocked Pipeline Stages), se conoce como MIPS a toda una familia de microprocesadores de arquitectura RISC desarrollados por MIPS Technologies.

El origen de la arquitectura MIPS se remonta al año 1981. En la universidad de Stanford, donde un equipo liderado por John L. Hennessy comienza a trabajar con lo que sería el primer procesador MIPS. Con la idea de mejorar a gran escala el rendimiento de la máquina a través del uso de la segmentación .

Sintetiza las principales características de la arquitectura RISC, es una arquitectura simple y eficiente.

Un procesador MIPS es toda una familia de microprocesadores de arquitectura RISC. Es utilizado en Windows CE, routers Cisco, en la Nintendo 64, PlayStation, PlayStation 2 y PlayStation Portable.

***Organización de un MIPS***

* Unidad Aritmética y Lógica (ALU).
* Unidad Aritmética entera, operaciones de multiplicación y división.
* Unidad Punto Flotante.
* Coprocesador dedicado al manejo de memoria caché y virtual.

La arquitectura MIPS requiere que el software implemente algunas limitaciones en el diseño que está normalmente considerado parte de la implementación del hardware. Este papel presenta resultados experimentales en la efectividad de este procesador como un programa anfitrión.

***Registros***

Tiene 32 registros de 32 bits de propósito general (GPR)

32 registros de 32 bits de punto flotante (FPR)

Cuenta con instrucciones diferenciadas para GPR y FPR

***Tipos de datos***

Half Word (16 bits)

Word (32 bits)

Double Word (64 bits)

Simple precisión (32 bits)

Doble precisión (64 bits)

Los datos half Word y Word se cargan en GPRs y se completan con 0 o el signo

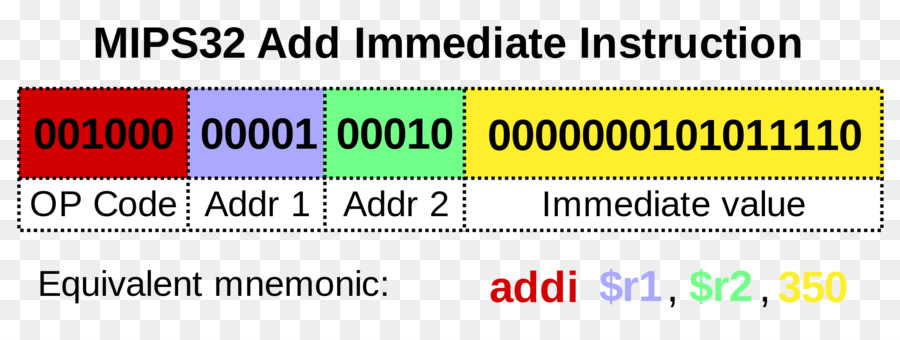
Las direcciones de memoria para cada instrucción son múltiplos de 4.

**SET DE INSTRUCCIONES**

Es una especificación que detalla las instrucciones que una CPU puede entender y ejecutar, o el conjunto de todos los comandos implementados por un diseño particular de una CPU. El término describe los aspectos del procesador generalmente visibles para un programador, incluidos los tipos de datos nativos, las instrucciones, los registros, la arquitectura de memoria y las interrupciones, entre otros aspectos.

Existen 3 principales tipos de set de instrucciones: CISC, RISC y SISC.

La arquitectura del conjunto de instrucciones (ISA) se emplea a veces para distinguir este conjunto de características de la microarquitectura, que son los elementos y técnicas que se emplean para implementar el conjunto de instrucciones. Entre estos elementos se encuentras las microinstrucciones y los sistemas de caché.



***TIPOS DE INSTRUCCIONES MIPS***

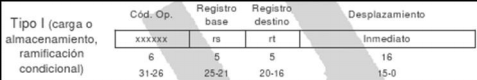
***Formato R***

Utilizado por las instrucciones aritméticas y lógicas



***Formato I***

Utilizado por las instrucciones de transferencia, las de salto condicional y las instrucciones con operando inmediatos



***Formato J***

Utilizado por las instrucciones de bifurcación



**TABLA DE INSTRUCCIONES MIPS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Instrucción* | *Tipo* | *Sintaxis* | *Descripción* | *Tabla Info* |
| Add | **R** | **Add $rd, $rs, $rt** | Suma Palabra (Add Word) | **∞** |
| Addi | **I** | **Addi $rd, $ro1, $inm** | Suma Palabra Inmediata (Add Word Inmediate) |
| Sub | **R** | **Sub $rd, $ro1, $ro2** | Resta palabra (Subtract Word) | **ℓ** |
| Mul | **R** | **Mul $rd, $ro1, $ro2** | Multiplica palabra (Multiply Word) | **Ω** |
| Div | **R** | **Div $ro1, $ro2** | Divide Palabra (Divide Word) | **∂** |
| Or | **R** | **Or $rd, $ro1, $ro2** | O (Or) | **∆** |
| Ori | **I** | **Ori $rd, $ro1, $inm** | O Inmediato (Or Inmediate) |
| And | **R** | **And $rd, $ro1, $ro2** | Y (And), Pone el registro | **∑** |
| Andi | **I** | **Addi $rd, $ro1, $inm** | Y Inmediato (And inmediate) |
| slt | **R** | **slt $rd, $ro1, $ro2** | Activa si menor (Set on Less Than) | **∫** |
| Slti | **I** | **slti $rd, $ro, $inm** | Activa si menor inmediato (Set on Less Than Inmediate) |
| not | **R** | **not $rd, $ro** | No (Not) | **∩** |
| Lw | **I** | **Lw $rd, $dir** | Carga Palabra (Load Word) | **ⱴ** |
| Sw | **I** | **Sw $ro, $dir** | Almacena Palabra (Store Word) | **ⱷ** |
| beq | **I** | **beq $ro1, $ro2, $etiq** | Bifurcación si igual (Branch on equal) | **●** |
| bne | **I** | **bne $ro1, $ro2, $etiq** | Bifurcación si no igual (Branch on not equal) |
| bgtz | **I** | **bgtz $ro, $etiq** | Bifurcación si mayor que cero (Branch on greater or equal than zero) | **○** |
| j | **J** | **j $etiq** | Salto (Jump), | **◊** |

|  |  |
| --- | --- |
| Simbología | **Significado** |
| ∞ | Suma el valor del registro ro1 y el registro ro2 (o el valor inmediato de 16 bits inm) y deja el resultado en el registro rd. Se consideran los operandos en complemento a 2, y cuando se produce un desbordamiento no se modifica el registro rd y se provoca una excepción del procesador |
| ℓ | Pone en el registro rd el resultado de calcular la resta entre los valores de los registros ro1 y ro2. Si la instrucción se consideran los operadores en complemento a 2, y cuando se produce desbordamiento no se modifica el registro rd y se provoca una excepción del procesador |
| Ω | Multiplica el valor de los registros ro1 y ro2 y deja los 32 bits menos significativos del resultado en el registro rd. Las instrucciones mul realizan multiplicación en complemento a 2. |
| ∂ | Divide el valor del registro ro1 entre el valor del registro ro2. Ambos valores se consideran cantidades enteras con signo. El cociente de la división se almacena en el registro L0 y el resto de la división en el registro HI. Esta instrucción nunca provoca una excepción |
| ∆ | Pone en el registro rd el resultado de realizar una operación or a nivel de bit entre los registros ro1 y ro2. En el caso de la instrucción ori entre el registro ro y el valor de 16 bits inm extendido con ceros |
| ∑ | Pone en el registro rd el resultado de realizar una operación *and* a nivel de bit entre los registros ro1 y ro2 (en caso de la instrucción and) o entre el registro ro y el valor de 16 bits inm extendido con ceros (Instrucción Andi) |
| ∫ | Pone el valor 1 en el registro rd, si los valores de los registros ro1 y ro2 cumplen la correspondiente condición. En otro caso pone el registro rd a cero. Si la comparación implica algún tipo de orden se considera que los números se interpretan en complemento a 2. |
| ∩ | Pone el registro rd el resultado de realizar una operación not a nivel de bit sobre el registro ro |
| ⱴ | Carga el valor de 4 bytes almacenado a partir de la posición de memoria dir en el registro rd. Si la dirección dir no está alineada a nivel de palabra (dirección múltiplo de 4) se produce una excepcion |
| ⱷ | Almacena a partir de la posición de memoria dir el valor de 4 bytes que se encuentra en el registro ro. Si la dirección dir no está alineada a nivel de palabra (dirección multiplo de 4) se produce una excepción. |
| ● | Salta, si los valores de los registro ro1 y ro2 cumplen una relación, a la instrucción marcada con etiq |
| ○ | Salta, si el valor del registro ro cumple una relación con respecto al valor cero, a la instrucción marcada con etiq |
| ◊ | Salta a la dirección marcada con etiq |

**DESARROLLO DE DATA PATH**

Una Data Path es una colección de unidades funcionales como unidades lógicas aritméticas o multiplicadores que realizan operaciones de procesamiento de datos, registros y buses.

Junto con la unidad de control, compone la unidad central de procesamiento (CPU). Se puede crear una ruta de datos más grande uniendo más de una ruta de datos utilizando multiplexores. Una ruta de datos es la ALU, el conjunto de registros y los buses internos de la CPU que permiten que los datos fluyan entre ellos.

Una ruta de datos de microarquitectura organizada alrededor de un solo bus, el diseño más simple de una CPU utiliza un bus interno común. La utilización eficiente requiere una estructura de tres buses internos un poco más complicada. Muchas CPU relativamente simples tienen un archivo de registro de 2 lecturas y 1 escritura conectado a las 2 entradas y 1 salida de la ALU.

A fines de la década de 1990, hubo una creciente investigación en el área de las rutas de datos reconfigurables (rutas de datos que se pueden reutilizar en tiempo de ejecución utilizando una estructura programable), ya que dichos diseños pueden permitir un procesamiento más eficiente y ahorros de energía sustanciales.

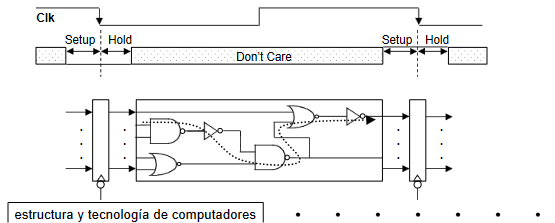
***Temporización Monociclo***

Ejecución típica (de una instrucción)

* Todos los registros se cargan simultáneamente (de modo selectivo)
* Todos los valores se propagan a través de las redes combinacionales hasta estabilizarse  
  en las entradas de los registros
* Se repite indefinidamente el proceso

Todos los elementos de almacenamiento están sincronizados al mismo flanco de reloj:

* Tiempo de ciclo = CKL-to-Q + Camino con retardo máximo + Setup + Clock Skew
* ( CLK-to-Q + Camino con retardo mínimo - Clock skew ) > Hold



**MEMORIA DE INSTRUCCIONES Y MEMORIA DE DATOS**

***Banco de Registros:***

|  |  |  |
| --- | --- | --- |
| Registro | Valor | Instrucción |
| *20* | *3* | *SUM* |
| *21* | *1* | *SUB* |
| *24* | *0* | *AND* |
| *25* | *10* | *OR* |
| *26* | *1* | *SLT* |
| *0* | *0* | *NOP* |

***Existe un error en una instrucción del archivo "TestF1\_MemInst.mem" debe identificarlo, anotarlo en el reporte y modificarlo para su correcto funcionamiento***

**REPORTE DE PROGRAMA EN VERILOG**