

Tema 4

Jerarquía de niveles estructurales

Código máquina: es código binario puro que se ejecuta en la CPU: difícil de manejar por el programador

Lenguaje ensamblador: es un lenguaje de programación de bajo nivel con correspondencia casi directa con el lenguaje máquina `add $7, $3, $6 # registro 7 ← registro 3 + registro 6`

Un computador funciona ejecutando continuamente instrucciones que operan sobre diferentes datos

Instrucciones y datos se almacenan en memoria y se codifican empleando un número específico de bits El computador está diseñado para transferir bloques de información de un determinado tamaño

- A cada uno de estos bloques se le denomina palabra
- El tamaño, en número de bits, de cada palabra se denomina ancho de palabra Todos los elementos del computador (registros, buses, memorias, periféricos, ALU) están diseñados para transferir, almacenar o procesar palabras

Ejecución de instrucciones

El proceso de ejecución de una instrucción sigue una secuencia de 4 pasos:

1. Lectura o carga de la instrucción: memoria → registro de instrucción (IR, Instruction Register).
2. Decodificación de la instrucción: La unidad de control lee el contenido del IR y decodifica la instrucción a ejecutar.
3. Ejecución de la instrucción: Se activan las señales de control necesarias para la ejecución de la operación.
4. Determinación de la siguiente instrucción: actualizando el contador del programa (PC, Program Counter).

El funcionamiento de la CPU está determinado por las instrucciones que ejecuta EL conjunto de instrucciones distintas que puede ejecutar se denomina repertorio de instrucciones El repertorio de instrucciones es la parte de la arquitectura de un computador relacionada con la programación y condiciona la implementación Distintos diseños de microprocesadores pueden tener el mismo repertorio de instrucciones (ejemplo: Intel Pentium y AMD Athlon) Cada instrucción debe contener toda la información necesaria para su ejecución

Instrucciones máquina

- Son las instrucciones que ejecuta directamente la CPU
- El lenguaje máquina (no el lenguaje ensamblador) es el nivel más bajo de programación. Es código binario Depende completamente del hardware
- Tipos de instrucciones:
 - De transferencias de datos: entre memoria y registros u operaciones con periféricos (E/S)
 - De operaciones aritméticas y lógicas
 - De control de flujo

- Instrucciones de bifurcación y salto condicional
- Instrucciones de llamada a subrutina
- De control del sistema y otras
- Los elementos constitutivos de una instrucción máquina son:
 - Código de operación
 - Referencia a operandos fuente: uno o más
 - Referencia al operando resuelto: si produce alguno
 - Referencia a la siguiente instrucción: en la mayoría de los casos es implícito y se refiere a la siguiente instrucción de la lista. En caso contrario, se suministrará la dirección de memoria.
- Los operadores fuente y resultado estarán en alguna de las siguientes localizaciones:
 - Memoria: deberá indicarse la dirección
 - Registro de la CPU: identificados por un número único
 - Dispositivos de E/S: módulo y dispositivo E/S o dirección de memoria en caso de E/S asignada en memoria
- La instrucción se divide en campos correspondientes a los elementos constitutivos de la misma (código de operación, operandos, etc.)
- La división en campos y bits se denomina formato de instrucción

4 bits	6 bits	6 bits
Codop	Ref. a operando	Ref. a operando

- 4 bits Codop
- 6 bits referencia a operando
- 6 bits referencia a operando
- 16 bits en total

- La mayoría de los repertorios de instrucciones emplean más de un formato
- Los aspectos más importantes en el diseño del repertorio de instrucciones son:
 - Repertorio de operaciones: Cuántas y qué operaciones considerar y cuán complejas deben ser.
 - Tipos de datos: Los distintos tipos de datos con los que se efectúan operaciones.
 - Formatos de instrucciones: Longitud de la instrucción (en bits), tamaño de los distintos campos, etc.
 - Registros: Número de registros de la CPU que pueden ser referenciados por instrucciones y su uso.
 - Direccionamiento: El modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

Estrategias de diseño

- Dos filosofías a la hora de diseñar un repertorio de instrucciones: CISC (Complex Instruction Set Computer)
 - Juego de instrucciones muy rico
 - Modos de direccionamiento potentes y complejos

- Programas con menos instrucciones, pero de ejecución compleja
 - RISC (Reduces Instruction Set Computer)
- Número relativamente pequeño de instrucciones disponibles
- Sencillas y con pocos modelos de direccionamiento
- Programas con más instrucciones, pero de ejecución muy eficiente
- Estudiaremos el repertorio del MIPS (Microprocessor without Interlocked Pipeline Stages) Diseño pionero RISC presentado en 1981 Usado durante años en supercomputadores y que en la actualidad se utiliza fundamentalmente en sistemas empujados Empleado por NEC, Nintendo, SGI, Sony, ...

Modelo de ejecución

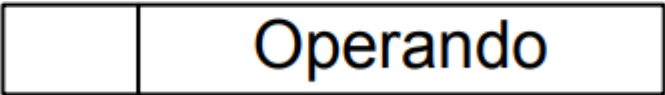
- El modelo de ejecución especifica el dispositivo (memoria, registro, etc.) que almacena los operandos de las instrucciones.
- Una misma arquitectura suele dar soporte a varios modelos de ejecución
- Los modelos soportados dependen fuertemente de la arquitectura.
 - Por ejemplo una ALU exclusivamente conectada con registros quedará limitada al modo registro-registro.
- Los modelos de ejecución posibles son:
 - Modelo de pila: POP y PUSH
 - Modelo registro-registro: el modo más rápido, Se necesita - transferir los datos.
 - Modelo registro-memoria: un operador en registro, otro en memoria. EL resultado se almacena en un registro.
 - Modelo memoria-memoria: No se necesita transferir los datos. La ejecución requiere un alto número de ciclos.

Modos de Direccionamiento

- Son los algoritmos empleados por el procesador para calcular las direcciones de las instrucciones y datos.
- La dirección real de memoria especificada por el modo de direccionamiento se denomina dirección efectiva.
- Modos de direccionamiento:
 - Direccionamiento inmediato.
 - Direccionamiento directo.
 - Direccionamiento indirecto.
 - Direccionamiento de registro.
 - Direccionamiento indirecto con registro.
 - Direccionamiento con desplazamiento:
 - Desplazamiento relativo
 - Desplazamiento con registro-base.
 - Indexado

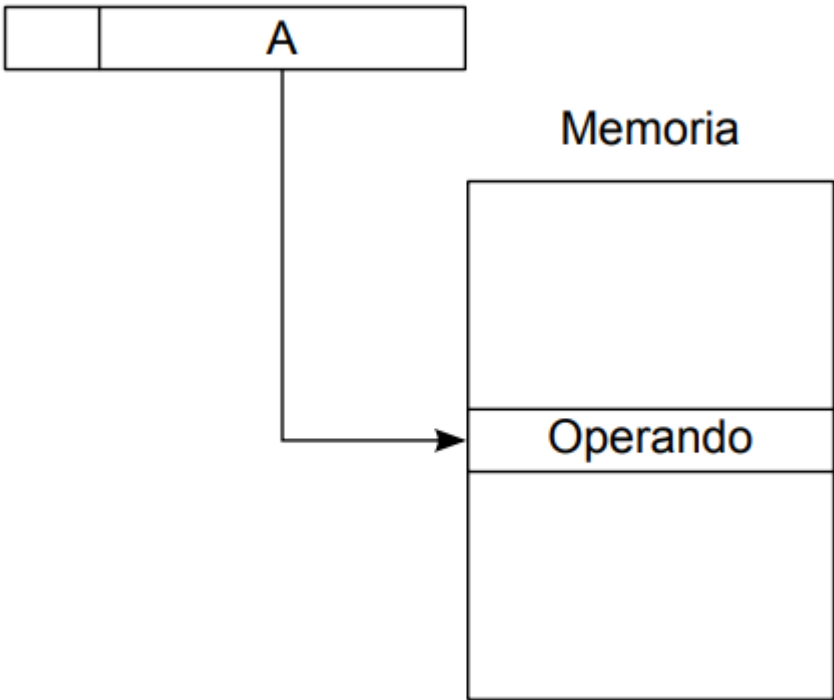
Modos de Direccionamiento

Instrucción



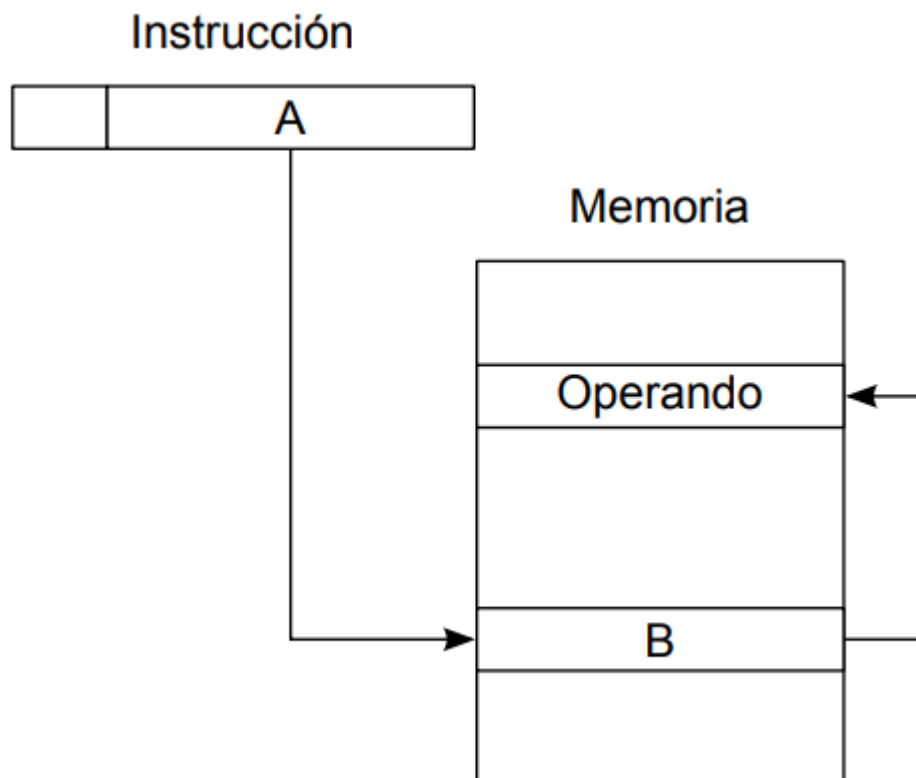
- El operando es una constante cuyo valor se almacena en el campo operando de la instrucción.
- Ventaja: no se requiere una referencia a memoria para obtener el operando. Desventaja: el tamaño del número está limitado por la longitud del campo de direcciones.

Instrucción



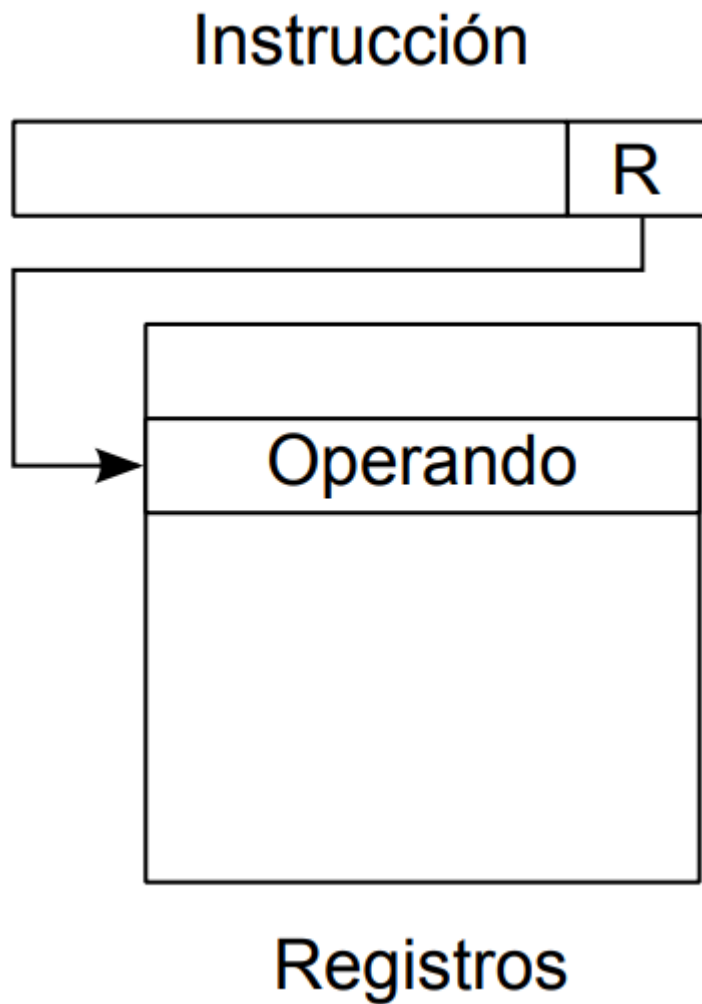
- El opereando se encuentra almacenado en memoria en la dirección indicada por el campo operando.

- Limitación: proporciona un espacio de direcciones reducido



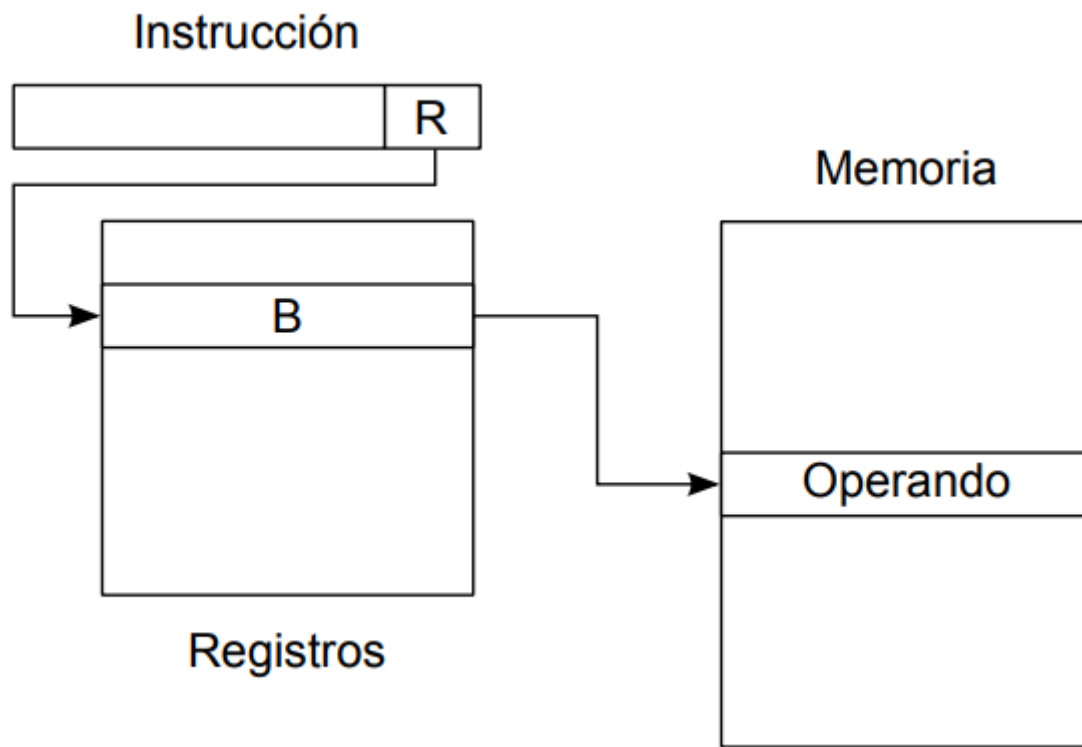
- El campo operando de la instrucción contiene la dirección a una posición de memoria que a su vez contiene la dirección del operando deseado.

- Desventaja: la ejecución de una instrucción requiere dos referencias a memoria principal.



- El operando referenciado se encuentra en un registro.
- El número de registro se especifica en el campo operando de la instrucción.
- Ventajas:
 - Sólo es necesario un campo pequeño de direcciones en la instrucción.
 - El tiempo de acceso a un registro es mucho menor que a memoria.
- Desventaja: espacio de direcciones muy limitado.

Direccionamiento indirecto con registro

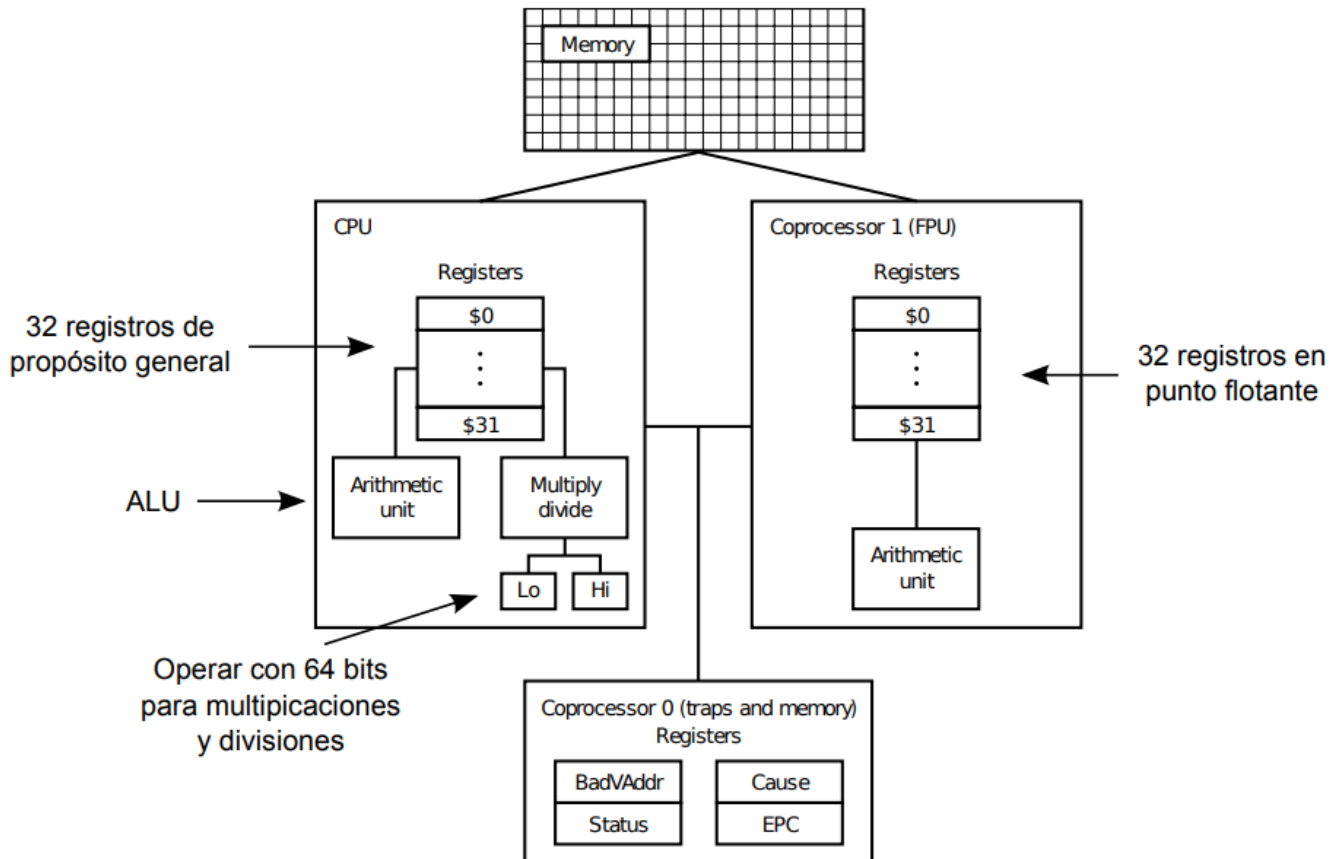


- El operando de la instrucción contiene un número de registro que contiene la dirección de memoria del operando deseado.
- Ventaja: Emplea una referencia a memoria menos que el direccionamiento indirecto.

Direccionamiento con desplazamiento

- El campo operando contiene una dirección relativa o desplazamiento(D).
- La instrucción especifica otra dirección (usualmente un registro) donde se almacena información del direccionamiento.
- La dirección efectiva se calcula como $(D+R)$.
- Desplazamiento relativo:
 - El registro referenciado (implícitamente) es el contador de programa.
 - La dirección efectiva es por tanto un desplazamiento relativo a la dirección de la instrucción.
- Direccionamiento con registro-base:
 - El registro referenciado (implícita o explícitamente) contiene una dirección de memoria.
 - El campo de dirección contiene un desplazamiento positivo o negativo desde esa dirección.
- Indexado:
 - El operando referencia una dirección de memoria.
 - El registro referenciado contiene un desplazamiento positivo desde esa posición.

Arquitectura del MIPS R2000



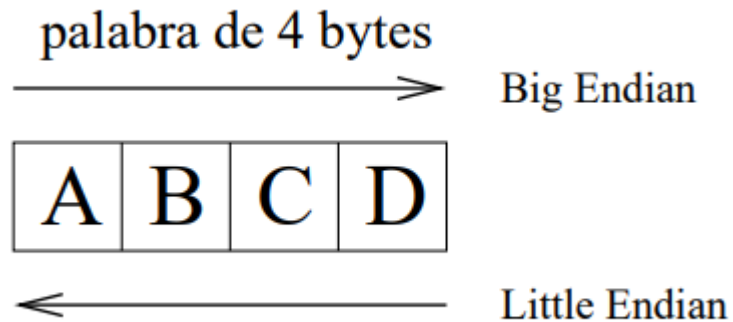
Características

- Longitud de palabra fija de 32 bits
- Direcciona bytes individuales: para indicar la dirección de una palabra en memoria hay que dar la dirección del primer byte
 - 2 palabras consecutivas estarán separadas en 4 unidades
 - En las operaciones de transferencia entre memoria y registro las direcciones de memoria han de ser múltiplos de 4: **resticción de alineamiento**
 - Las direcciones múltiplos de 4 se llaman direcciones alineadas
 - El intento de acceso a una unidad mayor de un byte en una dirección no alineada produce **un error de bus**

Orden de los bytes en una palabra

- Representación de los bytes en una palabra
 - **Big-endian:** la dirección del dato es el byte más significativo del mismo.
 - **Little-endian:** la dirección del dato es el byte menos significativo del mismo

- **Bi-endian**: configurable (MIPS)



Tipos de instrucciones MIPS

- Tipos de instrucciones soportados:
 - Transferencia de datos
 - Aritméticas y lógicas
 - De control de flujo:
 - Salto condicional
 - Bifurcación
- Características:
 - Longitud de todas las instrucciones fija de 32 bits.
 - Los operandos de las operaciones aritméticas son siempre registros
 - EL acceso a memoria se hace a través de operaciones de carga/almacenamiento (transferencia de datos).

Formatos de las instrucciones MIPS

Tipo - R

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Tipo - I

op	rs	rt	dirección/inmediato
6 bits	5 bits	5 bits	16 bits

Tipo - J

op	dirección
6 bits	26 bits

Instrucciones aritmético-lógicas:

- add, sub, or, and, ...
- Formato tipo R

- Direcccionamiento de registro

add \$7,\$3,\$6

0	3	6	7	0	32
---	---	---	---	---	----

000000	00011	00110	00111	00000	100000	
31	25	20	15	10	5	0

sub \$7,\$3,\$6

0	3	6	7	0	34
---	---	---	---	---	----

000000	00011	00110	00111	00000	100010	
31	25	20	15	10	5	0

Operaciones aritméticas con operandos inmediatos:

- addi, andi, ori, ...
- Formato tipo I
- Direcccionamiento de registro + direccionamiento inmediato

add \$7,\$3,\$6

0	3	6	7	0	32
---	---	---	---	---	----

000000	00011	00110	00111	00000	100000	
31	25	20	15	10	5	0

sub \$7,\$3,\$6

0	3	6	7	0	34
---	---	---	---	---	----

000000	00011	00110	00111	00000	100010	
31	25	20	15	10	5	0

Operaciones de transferencia:

- lw (load word) y sw (store word)
- Tipo I
- Direccionamiento con desplazamiento

lw \$8,1200(\$15)

35	15	8	1200
----	----	---	------

100011	01111	01000	0000 0100 1011 0000
--------	-------	-------	---------------------

31 25 20 15 0

sw \$8,1200(\$15)

43	15	8	1200
----	----	---	------

101011	01111	01000	0000 0100 1011 0000
--------	-------	-------	---------------------

31 25 20 15 0

Operaciones de transferencia con operandos inmediatos:

- lui (load upper inmediato)
- Tipo I
- Direccionamiento inmediato

Versión en lenguaje máquina: lui \$8, 255

001111	00000	01000	0000 0000 1111 1111
--------	-------	-------	---------------------

Contenido del registro \$8
tras la ejecución:

0000 0000 1111 1111	0000 0000 0000 0000
---------------------	---------------------

(\$rt=inm«16)

Instrucciones de salto condicional:

- beq (branch if equal) y bne (branch if not equal)
- Tipo I
- Direccionamiento con desplazamiento relativo
- beq \$rs, \$rt, despl salta a la instrucción que se encuentra en la dirección de memoria $(PC + 4) + 4 * \text{despl}$ si $\$rs == \rt .
- El ensamblador permite el uso de etiquetas para indicar la dirección de salto.
- Para hacer comparaciones del tipo *menor que* tenemos la instrucción lógica slt
- stl \$rd, \$rs, \$rt: $\$rf = (\$rs < \$rt)$
- stli \$rd, \$rs, inm: $\$rd = (\$rs < \text{inm})$

slt \$8,\$19,\$20

0	19	20	8	0	42
---	----	----	---	---	----

slti \$8,\$19,10

10	19	8	10
----	----	---	----

- slt: Tipo R, direccionamiento de registro
- slti: Tipo I, direccionamiento de registro + inmediato

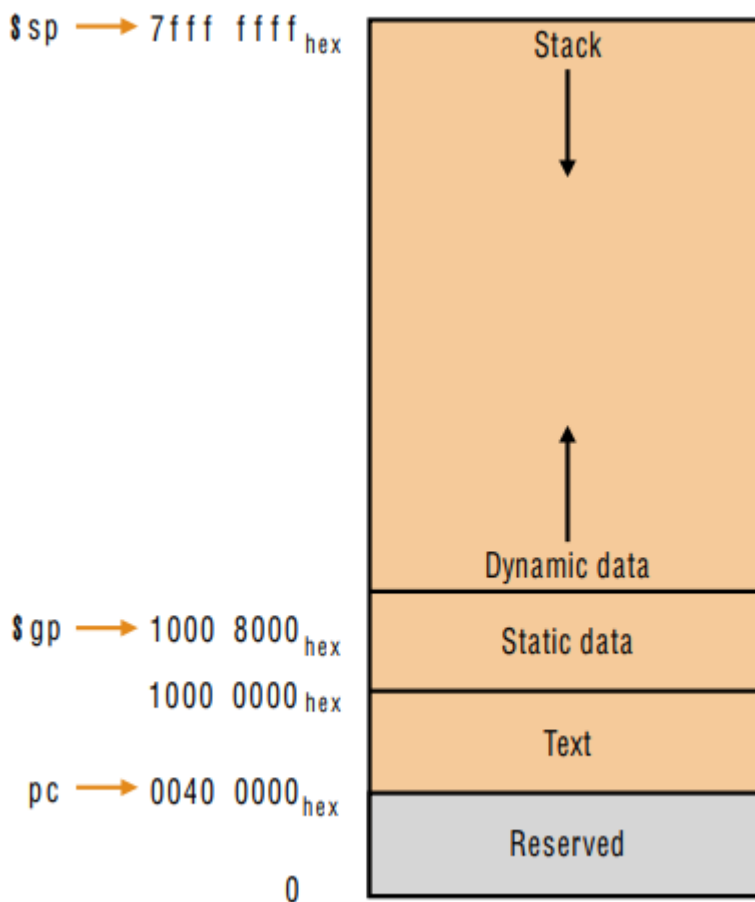
Instrucciones de bifurcación

- j (jump): Tipo J, direccionamiento pseudodirecto
 - $PC \leftarrow (PC + 4)_{31-28} \parallel R_{25-0}00$
 - El ensamblador permite el uso de etiquetas
- jr (jump register): Tipo R, direccionamiento indirecto con registro.

Llamadas a subrutinas

- Subrutina: es una secuencia de instrucciones de un programa que realiza una tarea específica y que está empaquetada como una unidad
- Es una herramienta para la estructuración de programas
 - Aumenta la legibilidad del código
 - Permite su reutilización.
- Pasos para la llamada a una subrutina:
 - Situar los parámetros en un lugar donde la subrutina pueda acceder a ellos.
 - Transferir el control a la subrutina. Adquirir los recursos de almacenamiento necesarios para el procedimiento. Realizar la tarea deseada. Sitúa el valor del resultado en un lugar donde el programa que lo ha llamado pueda acceder a él
 - Retornar el control al punto de origen

Distribución de memoria en MIPS



- Generalmente el invocador de la subrutina sitúa los parámetros en la pila.
- La subrutina colocará en la pila la información a guardar temporalmente.
- Al finalizar, liberará dicho espacio y dejará en la pila el resultado de la subrutina.
- Por último, el invocador recupera la pila el resultado.

Paso de parámetros a través de la pila

- Pila: espacio de memoria con estructura LIFO.
- El registro $\$sp$ apunta a la dirección más reciente utilizada.
- Crece de las direcciones de memoria superiores hacia las inferiores.
- Ejemplo de utilización
 - Push salvando los registros en la pila: `addi $sp, $sp, -8 sw $v0, 0($sp) sw $v1, 4($sp)`
 - Pop recuperando los datos de la pila a los registros: `lw $v0, 0($sp) lw $v1, 4($sp) addi $sp, $sp, 8`

Paso de parámetros a través de registros

- El acceso a los registros es más rápido que a memoria.
- El convenio de llamadas en MIPS se apoya en los registros para evitar accesos a pila:
 - $\$a0-\$a3$: paso de argumentos.
 - $\$v0-\$v1$: retorno de valores.
 - $\$ra$: dirección de retorno.
- Si se necesita pasar o devolver más argumentos, será necesario utilizar la pila.

Instrucción de salto

- jal (jump and link): Salta a una dirección y simultáneamente salva la dirección de retorno (\$ra = PC+4).
- Tipo J.
- Direccionamiento pseudodirecto. EL ensamblador permite el uso de etiquetas.
- EL salto de retorno se hará a través de jr \$ra.
- Uso:
 - El invocador pone los parámetros en \$a0-\$a3 y llama a jal.
 - La subrutina realiza los cálculos, pone los resultados en \$v0-\$v1 y devuelve el control
 - Si se necesita pasar más de cuatro argumentos o devolver más de dos valores se utilizará la pila.

Salvaguada de registros

- Si la subrutina modifica los registros utilizados por la rutina invocadora, los valores deberán ser guardados y restaurados antes y después, respectivamente, de la ejecución de la subrutina.
- Dos convenios estándar.
 - Guardar invocador (*caller save*).
 - Guardar invocado (*callee save*).
- Para evitar guardar y restaurar valores irrelevantes se usan las siguientes convenciones:
 - Registros temporales (\$t0-\$t9): no son preservados por el invocado. Es responsabilidad del invocador preservar su valor inicial si fuera necesario. Registros salvados (\$s0-\$s7): si el invocado los usa deberá salvar previamente su valor.
- La Unidad Central de Proceso (CPU) es la encargada de la ejecución de las instrucciones.
- La CPU se divide en dos bloques fundamentales:
 - Camino de datos: realiza las operaciones requeridas por las instrucciones máquina. Contiene:
 - Unidad aritmético-lógica.
 - Banco de registros.
 - Registros especiales: IR, PC, etc.
 - Buses internos: caminos de conexión entre los elementos de la CPU.
 - Unidad de control: gestiona el secuenciamiento de las operaciones ejecutadas en el camino de datos.
- La ejecución de una instrucción es un proceso dividido en 5 pasos
 1. **Captar la instrucción:** el procesador debe decodificarse para determinar qué acción es necesaria.
 2. **Interpretar la instrucción:** la instrucción debe decodificarse para determinar qué acción es necesaria.
 3. **Captar datos:** la ejecución puede exigir leer datos de la memoria o de un módulo de E/S.
 4. **Procesar datos:** la ejecución de una instrucción puede exigir llevar a cabo alguna operación aritmética o lógica.
 5. **Escribir datos:** los resultados de la ejecución pueden tener que ser escritos en la memoria o en un módulo de E/S.

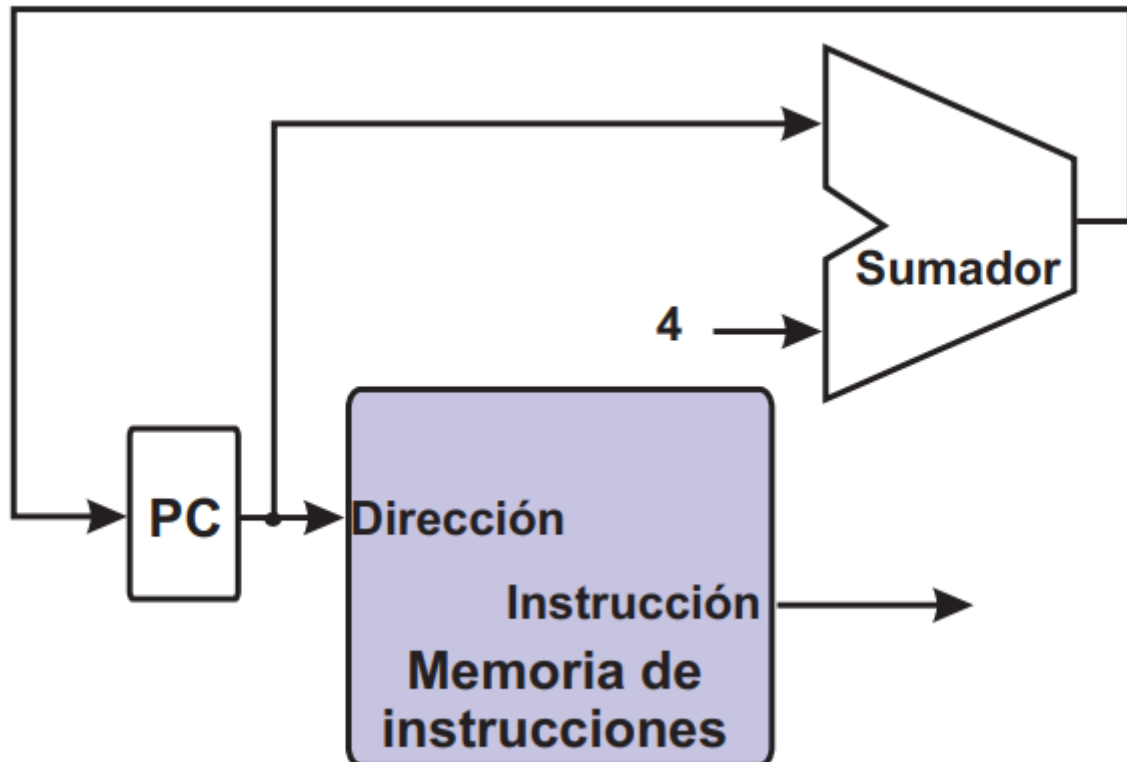
Introducción

- Estudiaremos un diseño sencillo de una CPU (camino de datos y unidad de control) capaz de ejecutar el siguiente subconjunto del repertorio de instrucciones MIPS:
 - Instrucciones de transferencia de datos: lw y sw.
 - Instrucciones aritmético-lógicas: add, sub, and, or y slt.

- Instrucciones de control de flujo: beq y j.
- Comenzaremos por la construcción del camino de datos
 - Utilizaremos módulos digitales combinacionales y secuenciales
 - Para los módulos secuenciales asumiremos sincronización por flancos.

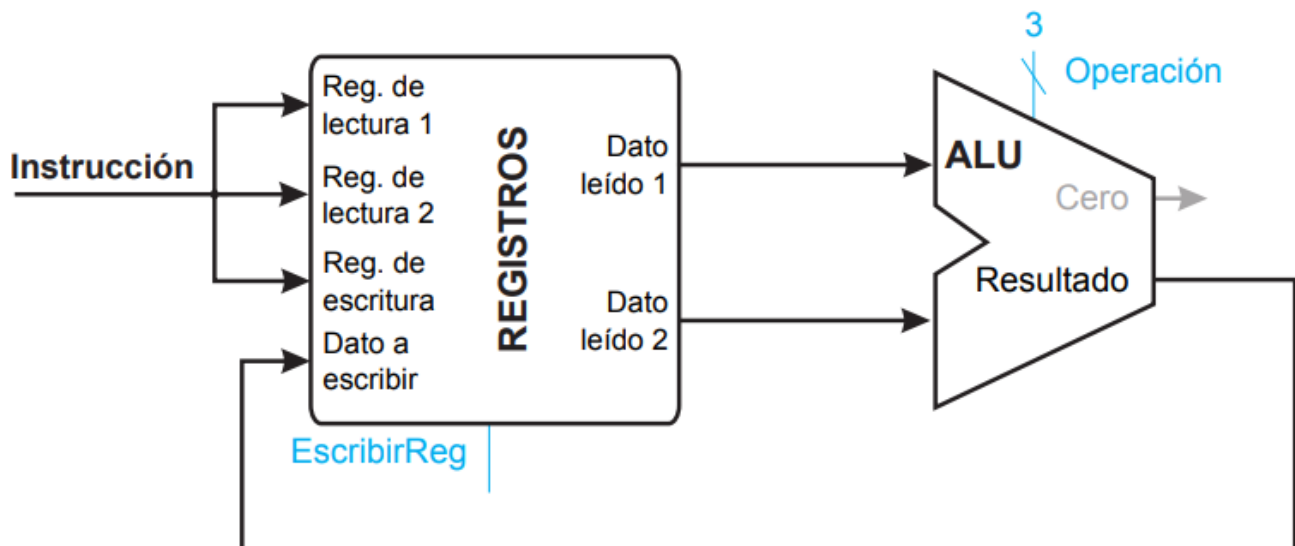
Construcción del camino de datos

Carga de la instrucción

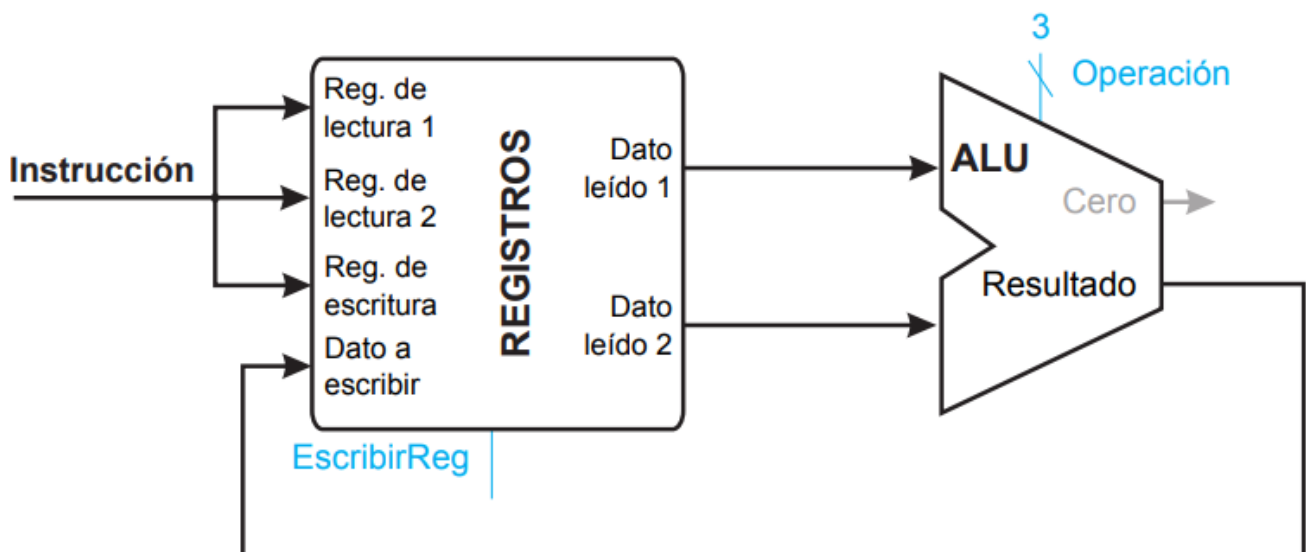


- La dirección de memoria en la que se localiza la instrucción a cargar la indica el PC.
- La siguiente instrucción se calcula incrementando el PC en 4 bytes.

Instrucciones aritmético-lógicas

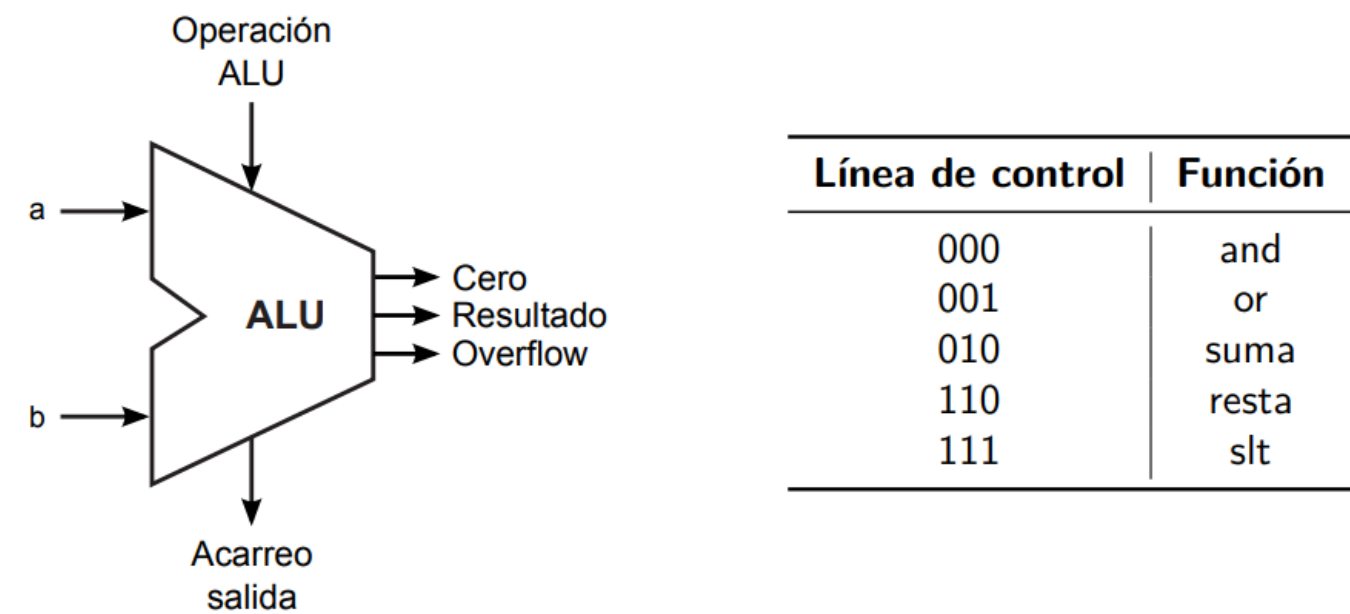


- Instrucciones tipo R que involucran tres registros: dos de lectura y uno de escritura.
- La operación se realiza a través de la ALU.
- Ejemplo típico: `add $t1, $t2, $t3`



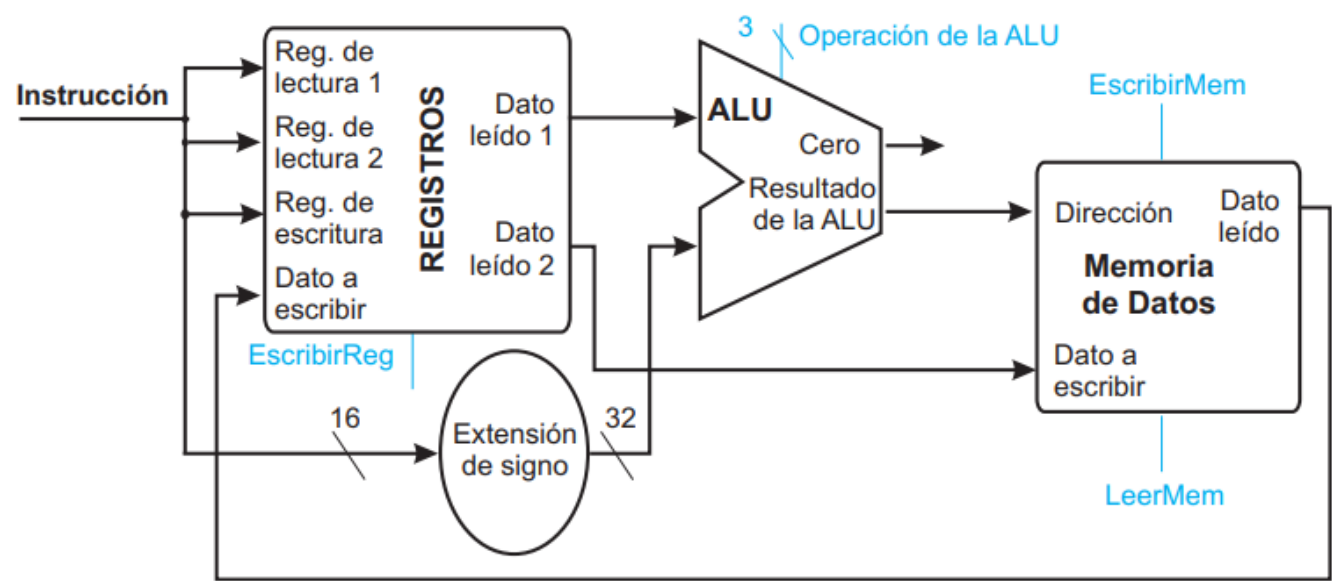
- Banco de registros:
 - Registros de 32 bits-
 - Dos puertos de lectura y uno de escritura.
 - Para escribir es necesario activar la señal **EscribirReg**.
 - Entradas (direcciones a leer/escribir): 5 bits.
 - Salidas: 32 bits (tamaño de palabra).

Unidad aritmético-lógica



La implementación de esta ALU se estudió en el tema de sistemas combinacionales

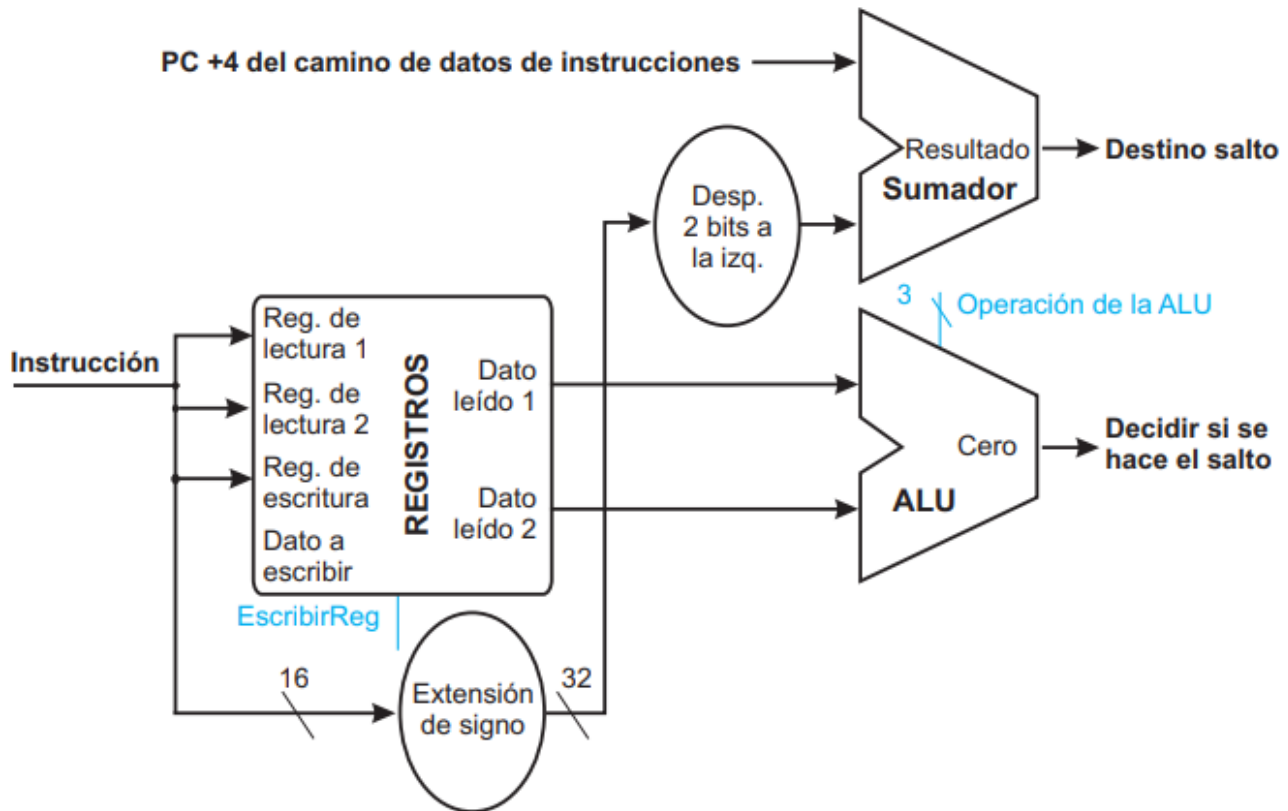
Instrucciones de transferencia de datos



- Instrucciones tipo I
- Modo de direccionamiento: registro-base
- Ejemplos típicos:
 - lw \$t1, desp1(\$t2)
 - sw \$t1, desp1(\$t2)
- Se utiliza la ALU para el cálculo de la dirección de memoria.
- El campo desplazamiento de la instrucción se extiende de 16 a 32 bits antes de realizar la suma.
- Memoria de datos:

- Entradas:
 - Señales de control de lectura y escritura.
 - Dirección a acceder (32 bits).
 - Palabra a escribir (32 bits).
- Salidas:
 - Palabra leída (32 bits).

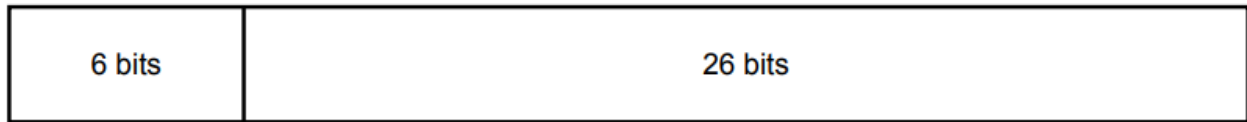
Instrucción de salto condicional



- Instrucción de tipo I: beq \$t1, \$t2, displ
- Evalúa si el contenido de \$t1 y \$t2 son iguales
- Si la condición se cumple se realiza el salto
- Comparación:
 - Se realiza una resta y se utiliza la salida Cero de la ALU como marcador de igualdad
- Dirección de salto:
 - Salto relativo al PC+4 con signo
 - Dirección de salto: $(PC+4) + 4 * \text{despl}$
 - Los 16 bits del desplazamiento se extiende a 32 y se desplazan 2 posiciones a la izquierda

Instrucción de salto incondicional

j L1

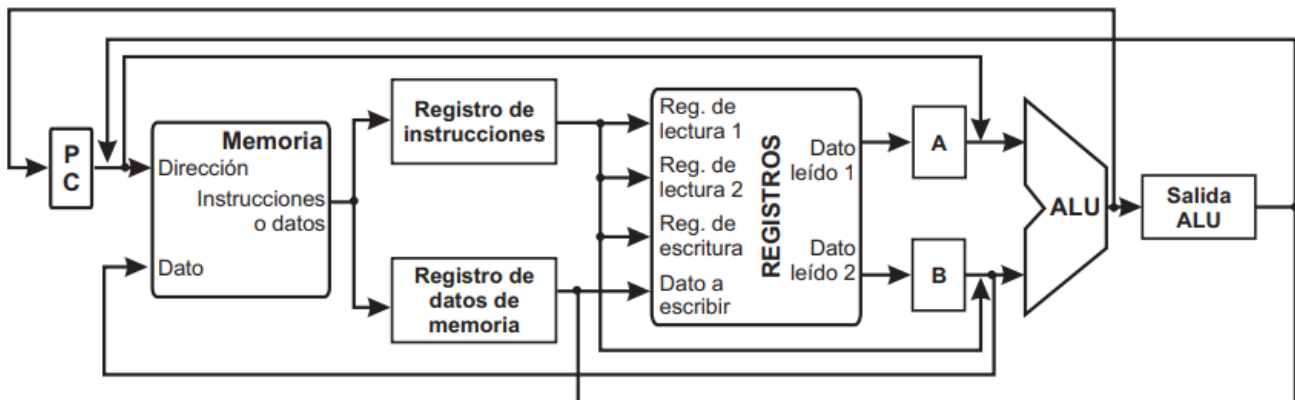


- Tipo J con direccionamiento pseudodirecto: j L1.
- Reemplaza los 28 bits de menor peso del PC con los 26 bits de menor peso de la instrucción desplazados 2 posiciones a la izquierda
- No es necesario la introducción de hardware adicional

Camino de datos completo

- Formado por la combinación de elementos necesarios por cada instrucción
- Cada una de las 5 etapas de ejecución de instrucciones se realiza en un ciclo de reloj diferente:
 - Se puede reutilizar una misma unidad funcional dentro de la misma instrucción si se utiliza en diferentes ciclos
 - No se necesita una memoria para instrucciones y otra para datos, ya que el acceso a instrucciones y datos es en ciclos diferentes
 - Para el cálculo de la dirección de la siguiente instrucción (PC+4 o dirección de salto) podemos usar la ALU en lugar de los sumadores
- Los datos que vayan a utilizarse en los siguientes ciclos deben almacenarse en un elemento de almacenamiento (memoria o registro)
 - Los datos a utilizar por las siguientes instrucciones en ciclos posteriores se almacenarán en elementos de almacenamiento visibles al programador (banco de registros, PC o memoria)
 - Los datos a utilizar por la misma instrucción en ciclos posteriores se almacenarán en registros temporales

Visión de alto nivel del camino de datos completo



- Una sola ALU en vez de 1 ALU y dos sumadores
- Una memoria única para instrucciones y datos

- Registros temporales tras cada unidad funcional para almacenar su salida hasta que ese valor se utilice en el siguiente ciclo
- Operaciones de la ALU:
 - Carga de la instrucción Suma $PC+4$.
 - Instrucción de transferencia de datos: cálculo de la dirección a acceder.
 - Instrucción tipo R: Operación aritmética o lógica.
 - Salto condicional:
 - Evaluación de la condición de salto.
 - Cálculo de la dirección de destino.
- Registros temporales:
 - Registro de instrucción (IR) y registro de datos de memoria (MDR):
 - Almacenan la instrucción o dato leído de memoria respectivamente.
 - Registros A y B: operandos leídos del banco de registros.
 - Salida ALU: salida de la ALU.
 - El registro IR requiere de una señal de control de escritura (necesita almacenar su valor más de un ciclo).
- Se introducen multiplexores para selección de entradas:
 - Origen de dirección de acceso a memoria:
 - Salida de la ALU (acceso a datos).
 - PC (acceso a instrucciones).
 - Registro de destino:
 - Instrucciones tipo R: rd (bits 11-15).
 - Instrucciones tipo I (cargas): rt (bits 16-20).
 - Dato a escribir:
 - De memoria (carga de datos).
 - ALU (instrucción aritmético-lógica).
 - Primera entrada de ALU:
 - registro A (acceso a datos e instrucción aritmético.lógica).
 - PC (carga de instrucción y saltos).
 - Segunda entrada de ALU:
 - Registro B (instrucción aritmético-lógica y salto condicional).
 - Constante 4 (incremento del PC).
 - Campo desplazamiento extendido (transferencia de datos).
 - Campo desplazamiento extendido y desplazado (saltos condicionales).
 - La dirección de la siguiente instrucción a ejecutar:
 - Resultado ALU: $PC+4$

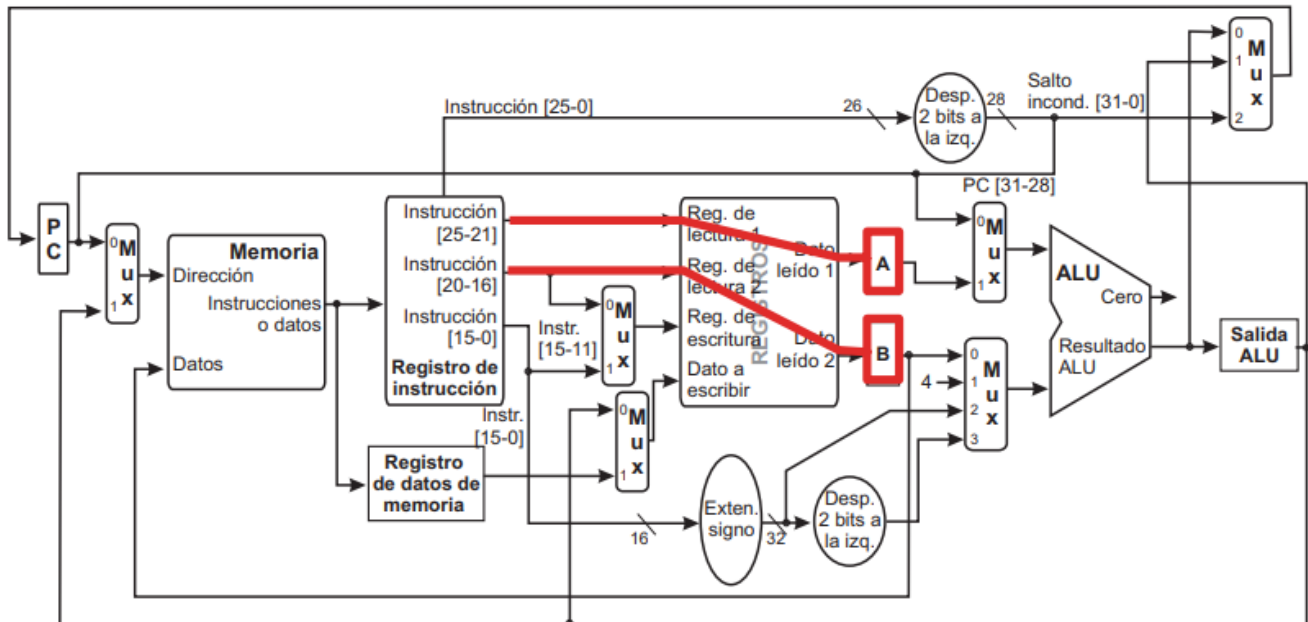
- [illegible]

El diagrama ilustra el flujo de datos en un procesador de instrucciones de 32 bits. Los componentes principales y sus conexiones son:

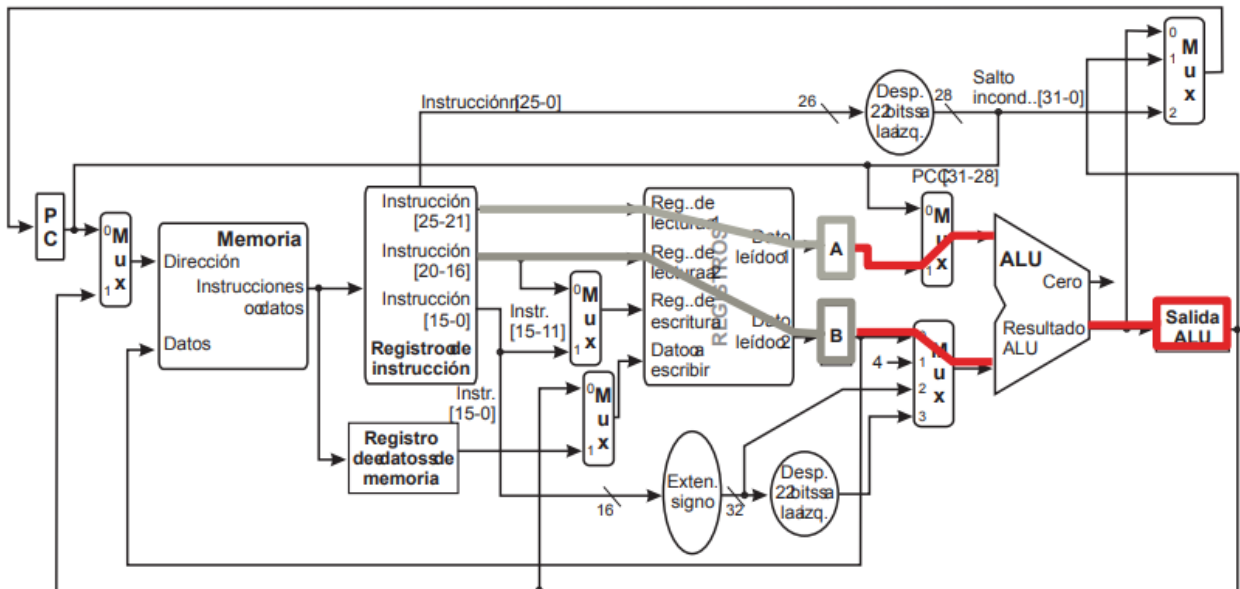
- PC (Program Counter):** Proporciona la dirección de la instrucción a la Memoria (Dirección Instrucciones o datos).
- Memoria:** Almacena instrucciones y datos. Proporciona la Instrucción [25-0] al Registro de Instrucción y los Datos al Registro de Datos de Memoria.
- Registro de Instrucción:** Almacena la instrucción actual. Se divide en:
 - Instrucción [25-21]: Dirección de la instrucción.
 - Instrucción [20-8]: Código de operación.
 - Instrucción [15-0]: Operando.
- Registro de Datos de Memoria:** Almacena los datos de la memoria. Proporciona los datos al Registro de Instrucción y al ALU.
- Registros:**
 - Reg. de lectura 1 y 2:** Almacenan los operandos de la instrucción.
 - Reg. de escritura:** Almacena el dato a escribir.
 - Reg. de resultados:** Almacena el resultado de la operación del ALU.
- ALU (Arithmetic Logic Unit):** Realiza operaciones aritméticas basadas en los operandos y el código de operación. El resultado se almacena en el Registro de Resultados.
- Salida ALU:** Proporciona el resultado de la operación del ALU al Registro de Resultados.
- PC [31-28]:** Proporciona la dirección de la siguiente instrucción al Registro de Instrucción.
- Desplazamiento (Desp.):** Proporciona el desplazamiento de la instrucción al Registro de Instrucción.
- Salto incond. [31-0]:** Proporciona la dirección de la siguiente instrucción al Registro de Instrucción.
- Extensión de signo (Ext. signo):** Proporciona el signo extendido de la instrucción al Registro de Instrucción.

El camino rojo indica la ejecución de una instrucción de salto incondicional. El PC se actualiza con la dirección de la siguiente instrucción (Salto incond. [31-0]) y el Registro de Instrucción se actualiza con la nueva instrucción.

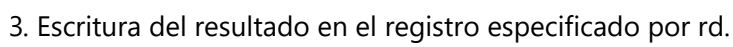
- 22 / 35



1. Lectura de los registros indicados por rs y rt y almacenamiento en los registros temporales A y B.

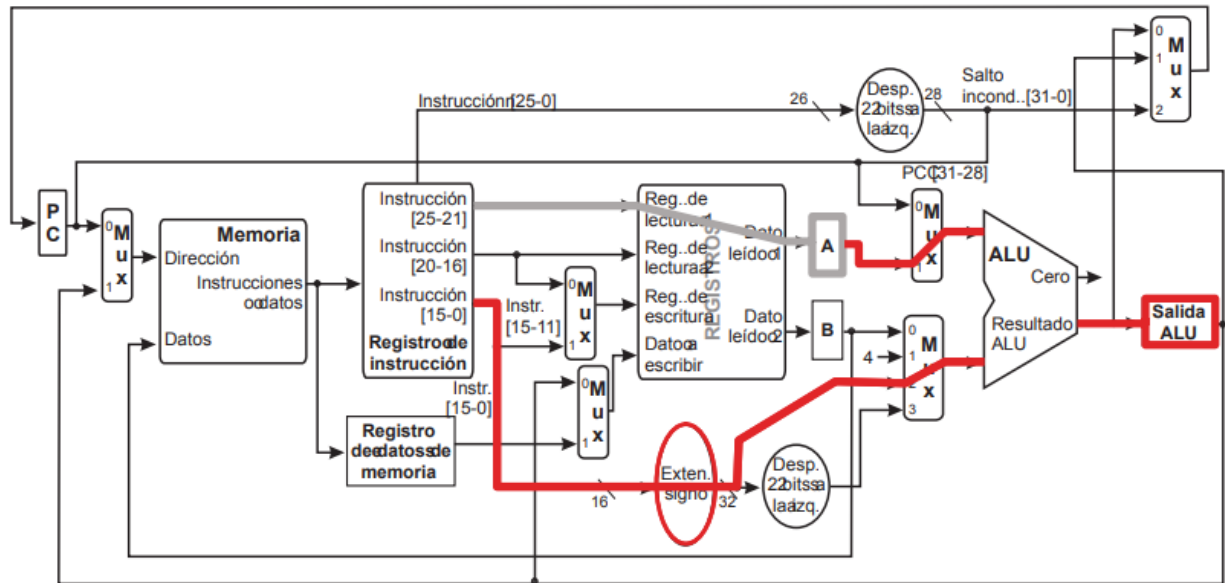


2. Ejecución por parte de la ALU de la operación especificada en el código de operación y almacenamiento del resultado en el registro de SalidaALU.

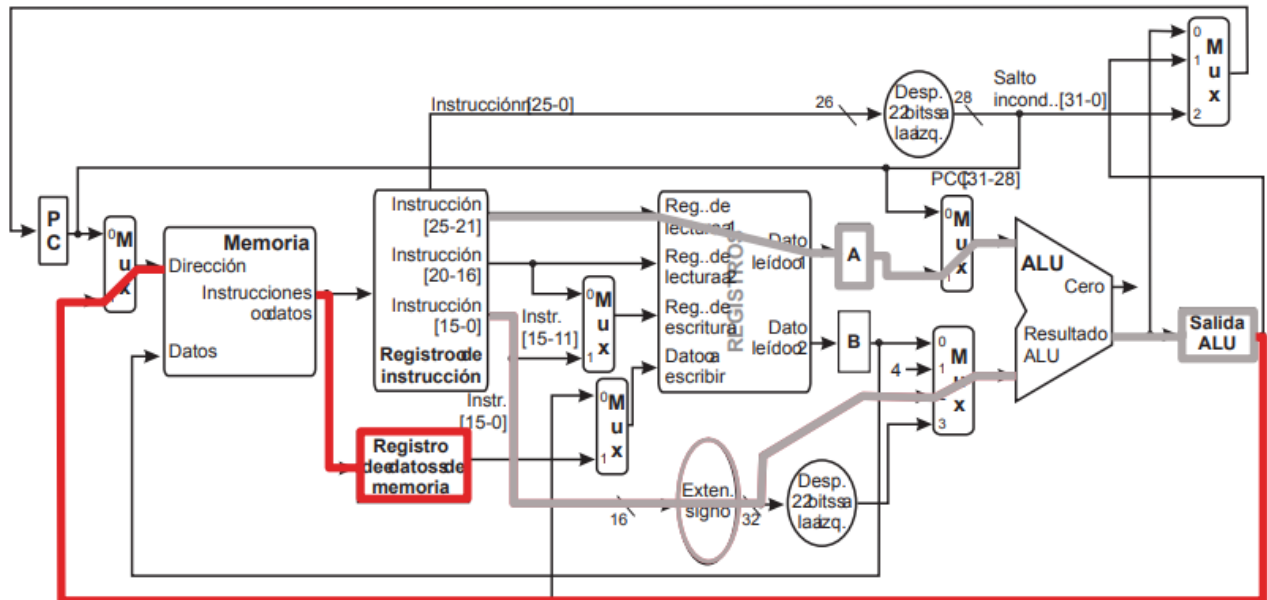


El diagrama ilustra el flujo de datos en un procesador de instrucciones de 32 bits. El PC (Program Counter) proporciona la dirección de la instrucción a la memoria y a un multiplexor. La memoria devuelve la instrucción y los datos. La instrucción es decodificada en campos como la dirección de salto incondicional y el signo extendido. Los datos de la memoria se extienden a 32 bits. El ALU realiza operaciones basadas en el resultado de cero y produce el resultado final de 32 bits.

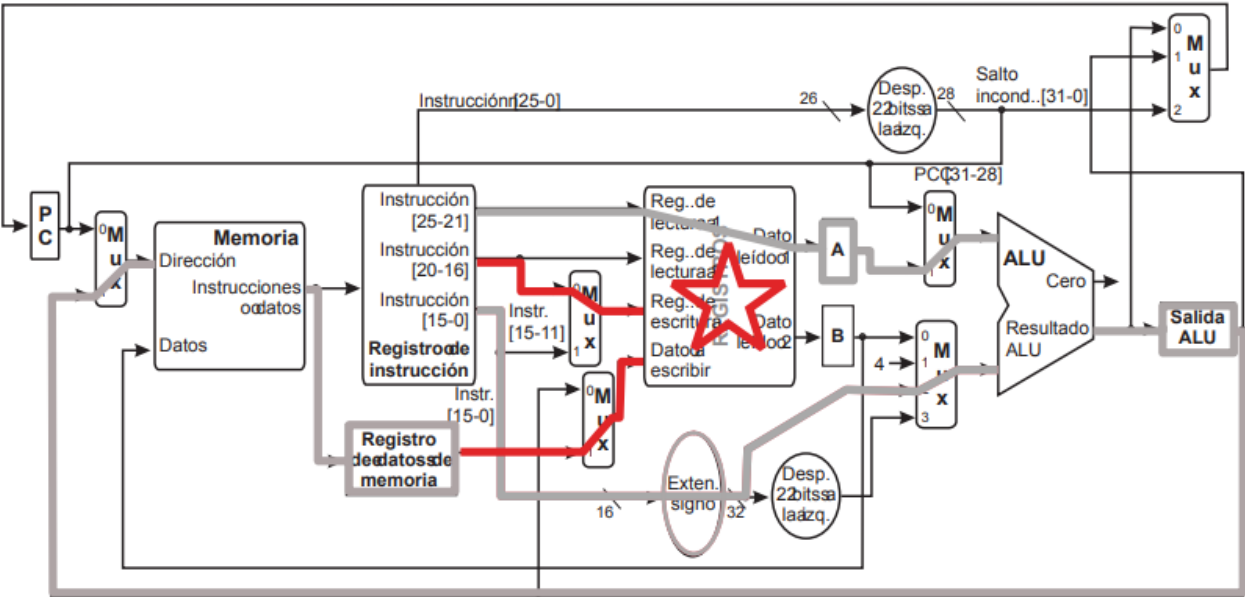
1. Lectura del registro base y almacenamiento en el registro temporal A.



2. Cálculo de la dirección de acceso como registro base más desplazamiento.

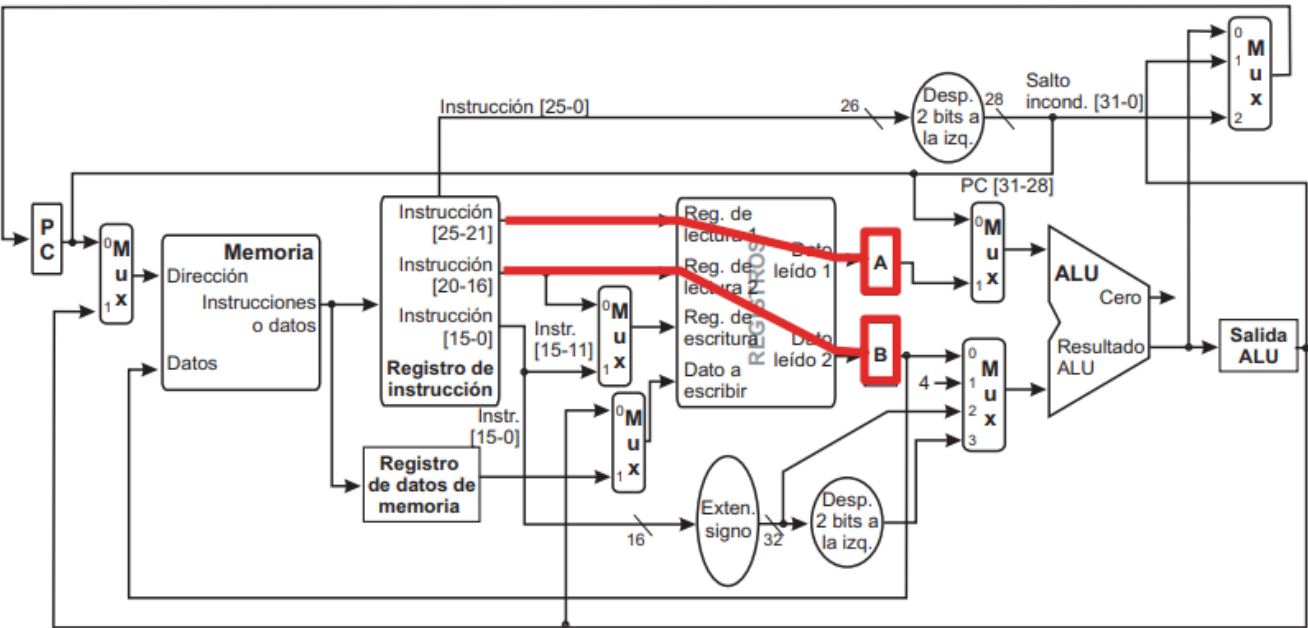


3. Acceso a memoria y carga del dato leído en el registro MDR.



4. Escritura del valor leído en blanco de registros,

Instrucción sw



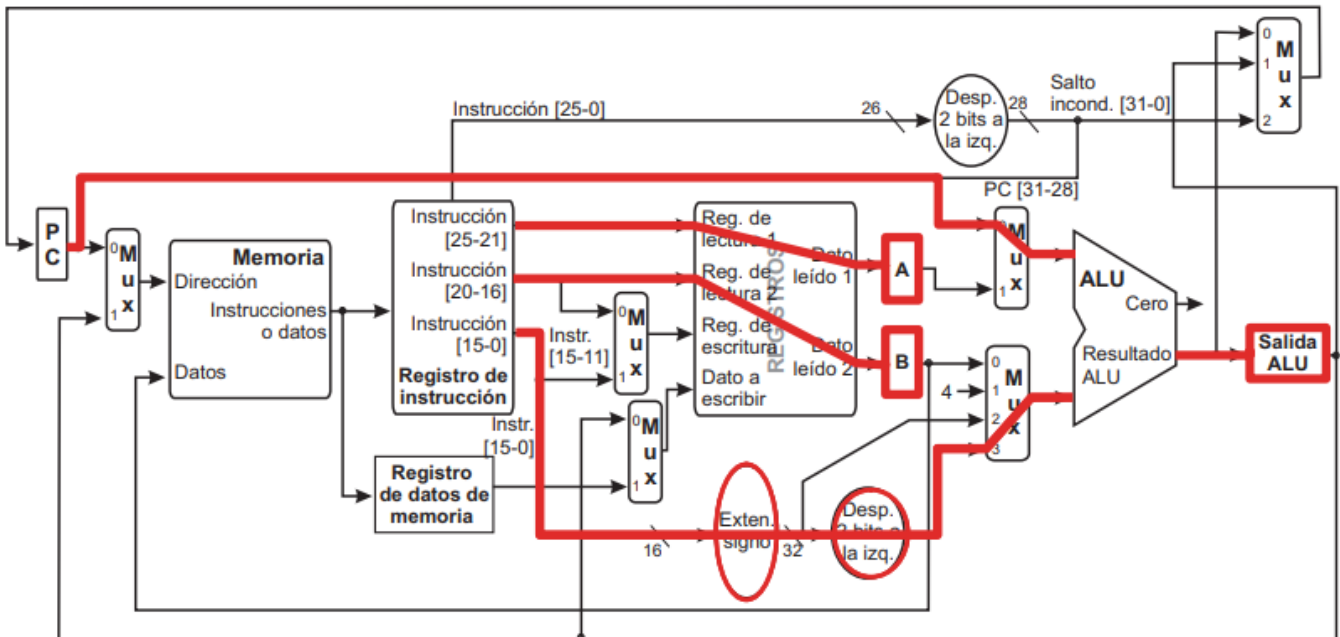
El diagrama ilustra el procesamiento de una instrucción MIPS en modo de ejecución. Los componentes principales y sus conexiones son:

- PC (Program Counter):** Proporciona la dirección de la instrucción a la memoria y a un multiplexor de 2 bits.
- Memoria:** Recibe direcciones de instrucciones y datos. Proporciona la instrucción completa y los datos de la instrucción.
- Registros de Instrucción:** Dividen la instrucción en campos: Instrucción [25-21], Instrucción [20-16], Instrucción [15-0], y Registro de instrucción.
- Registros de Datos:** Almacenan los datos de la instrucción.
- Multiplexores (MUX):** Seleccionan los registros de destino y fuente basados en los campos de la instrucción.
- ALU:** Realiza operaciones aritméticas y lógicas basadas en los operandos seleccionados y el código de operación.
- Salida ALU:** Proporciona el resultado de la operación.
- Extensión de signo:** Extiende el signo de los operandos de 16 bits a 32 bits.
- Salto incondicional (jalr):** Se resalta en rojo el camino de datos para esta instrucción, que utiliza el registro de destino y el registro de datos para calcular la nueva dirección del PC.

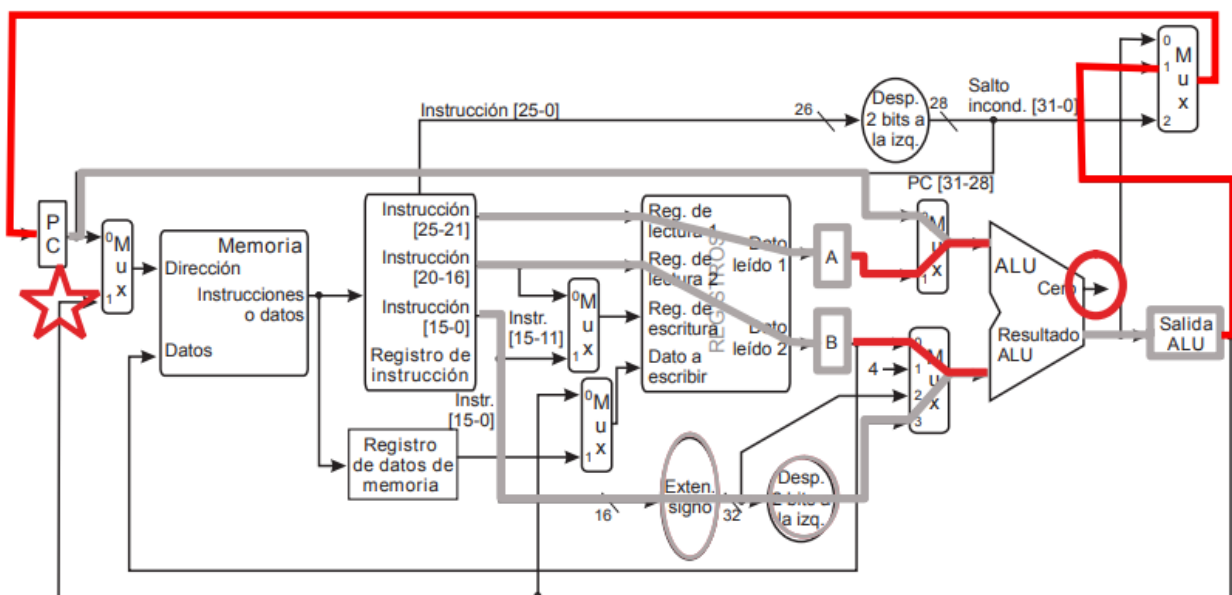
El diagrama ilustra el procesamiento de una instrucción MIPS en modo R. El flujo principal es el siguiente:

- PC (Program Counter):** Proporciona la dirección de la instrucción a la memoria.
- Memoria:** Entrega la instrucción completa (26 bits) y los datos (32 bits) a los registros de instrucciones y datos, respectivamente.
- Registros de Instrucción:** Se dividen en tres partes:
 - Instrucción [25-21]:** Selecciona el registro de lectura (Reg. de lectura).
 - Instrucción [20-16]:** Selecciona el registro de escritura (Reg. de escritura).
 - Instrucción [15-0]:** Selecciona el registro de datos (Reg. de datos).
- Multiplexores (MUX):**
 - El MUX de selección de registro de lectura (0-31) recibe la instrucción [25-21] y el registro de datos de la memoria.
 - El MUX de selección de registro de escritura (0-31) recibe la instrucción [20-16] y el registro de datos de la memoria.
 - El MUX de selección de registro de datos (0-31) recibe la instrucción [15-0] y el registro de datos de la memoria.
- Registros de Datos:** Almacenan los datos seleccionados por los MUX.
- ALU (Arithmetic Logic Unit):** Recibe los datos de los registros de lectura y escritura, y el desplazamiento de 26 bits (Desp. 26 bits laa.zq.) para operaciones de salto incondicional. El resultado de la ALU se almacena en el registro de datos.
- Salida ALU:** El resultado final de la operación.

27 / 35



1. Carga en paralelo de los operandos a comparar (registros temporales A y B). Al mismo tiempo, cálculo de la dirección de destino del salto.



2. Resta de los operandos. La señal de Cero de la ALU se utilizará para activar la escritura en el PC. Si se cumple la condición se escribe el PC con el nuevo valor.

Instrucción j

- El único paso a realizar es la escritura del PC con la dirección de salto.
- Serían precisos tan solo dos ciclos para su ejecución, pero como veremos se ejecutarán en 3 para simplificar la unidad de control

Señales de control

- Señales de control de escritura: PC, IR, memoria, banco de registros.
- Señal de lectura para la memoria.
- MUXes:

- 2 entradas: 1 línea de control.
- 4 entradas: 2 líneas de control.
- Unidad de control de la ALU:
 - Código de función + ALUOp = 3 bits de control.

Unidad de control de la ALU

- 00: Usado en instrucciones lw, sw y beq (para calcular el destino del salto).
- 01: Usando en beq para determinar la igualdad de los operandos.
- 10: En instrucciones tipo R, la operación a realizar depende del campo de función de la instrucción.
- ALUOp, SelALUB y Fuente PC son de 2 bits.
- El resto son de 1 bit.
- Los únicos registros que requiere señal de escritura son IR y PC.

Camino de datos y unidad de control

- Condiciones de escritura del PC:
 - Carga de la instrucción - Salto incondicional ($\text{EscrPC} = 1$).
 - Salto condicional con igualdad de operandos ($\text{EscrPCCond} = 1$ y $\text{Cero ALU} = 1$).

Diseño de la unidad de control

- La Unidad de Control debe especificar:
 - Las señales que se van a activar en cada paso
 - El paso siguientes de la secuencia
- Es un sistema secuencial síncrono:
 - Cada estado dura un ciclo de reloj y representa una etapa de la ejecución de una instrucción.
 - Entradas: los bits de la instrucción a ejecutar y los indicadores internos (p. ej. la salida Cero de la ALU). Salidas: las señales de control a activar.
- Los dos primeros estados son comunes a todas las instrucciones:
 - Carga de la instrucción.
 - Decodificación.
- El resto dependen del tipo de instrucción y sus operandos.
- Tras terminar la ejecución de una instrucción el autómata vuelve al estado inicial para procesar la siguiente instrucción

Etapas 2 - Decodificación y búsqueda de los registros

- Se decodifica la instrucción
- Se adelantan acciones que puedan ser útiles
 - Se leen los registros rt y rs y se almacenan en A y B.
 - Se calcula la dirección de salto potencial (beq).

Etapas 3 (lw o sw). Cálculo de la dirección de memoria

- SelALUA = 1: Entrada A de la ALU es el registro A (registro base).
- SelALUB = 10: Entrada B es el campo de desplazamiento con el signo extendido ALUop = 00: suma.

Etapa 4 (lw). Acceso a memoria

- LeerMem: Activar la memoria memoria para lectura.
- IoD = 1: Memoria direccionada por SalidaALU.
- El registro MDR no necesita señal de activación.

Etapa 5 (lw): Escritura del dato MDR al banco de registros

- EscrReg: Activar la escritura en el banco de registros.
- Mem2Reg = 1: El dato a escribir proviene del MDR.
- RegDest = 0: El registro destino proviene de los bits 16-20 de la instrucción (campo rt).

Etapa 4 (sw): almacenamiento del dato en memoria

- EscrMem: Activar la escritura en memoria
- IoD = 1 Memoria direccionada por SalidaALU.

Instrucciones de salto condicional (beq)

Etapa 3: Comparación y determinación del nuevo PC

- Comparación:
 - SelALUA = 1: Entrada A de la ALU es el registro A (rs).
 - SelALU = 00: Entrada B de la ALU es el registro B (rt).
 - ALUop = 01: Resta.
- Nuevo PC:
 - EscrPCCond: Activa la escritura del PC siempre y cuando, además la salida Cero de la ALU sea 1.
 - FuentePC = 01: El nuevo PC se encuentra en SalidaALU.

Instrucciones de salto incondicional (jump)

Etapa 3: Se escribe el nuevo PC

- EscrPC: Activar la escritura incondicional del PC.
- FuentePC = 10: El nuevo PC se obtiene tras desplazar dos bits a la izquierda los bits 0-25 de la instrucción, y concatenarlos con los 4 bits más significativos del PC.

Autómata de control complejo

- Dos técnicas diferentes de implementación:
 - Unidad de control cableada: se basa en alguno de los métodos clásicos de diseño
 - Unidad de control microprogramada: el autómata de control se representa en forma de programa de control
 - Más fácil de depurar, modificar y ampliar
 - Mejor opción para sistemas grandes y/o complejos

Control microprogramado

- El autómata se transforma en un programa de control
- Las instrucciones del programa de control se denominan microinstrucciones
- Cada microinstrucción define el conjunto de señales de control que se deben activar en un estado determinado
- El conjunto de las microinstrucciones se denomina microprograma
- El microprograma se almacenará en una memoria de solo lectura

2 formas:

- Existen 2 formas de realizar el secuenciamiento:
 - Explícito: cada microinstrucción incluye la dirección de la microinstrucción siguiente
 - Implícito: existen dos tipos de microinstrucción, de control y de salto.

Secuenciamiento

- Vamos a implementar la UC microprograma para el conjunto básico de instrucciones MIPS considerando secuenciamiento explícito
- Cada estado del autómata de control se transformará en una microinstrucción
- Elección de la siguiente microinstrucción, tres opciones:
 - Incrementar la dirección de la microinstrucción actual
 - Saltar a la primera microinstrucción para iniciar la ejecución de una nueva instrucción
 - Elegir la siguiente microinstrucción en función del estado en el que nos encontramos y la instrucción que estamos ejecutando
 - Tabla de envío: tabla con las direcciones destino
 - Una tabla para cada estado con múltiples estados destino
- Para la implementación de la UC del MIPS necesitamos dos tablas de envío
- Se necesitan 2 bits (CtrlDir) para especificar la siguiente microinstrucción
 - CtrlDir = 00: Salta a la primera microinstrucción para comenzar una nueva instrucción
 - CtrlDir = 01: Utiliza la tabla del envío 1
 - CtrlDir = 10: Utiliza la tabla del envío 2
 - CtrlDir = 11: Selecciona la siguiente microinstrucción

Implementación

- Cada punto de control del camino de datos se corresponde con un bit de las microinstrucciones
 - Cada microinstrucción consta de 18 bits
 - Los puntos de control podrían estar codificados para ahorrar bits

Temporización

Ciclo de reloj

	Actualizar registro (5 ns)	Acceso banco reg. (10 ns)	Leer memoria (25 ns)	ALU (15 ns)	Escribir memoria (30 ns)	Total
Carga instrucción	✓		✓			30 ns
Decodificación	✓			✓		20 ns
Cálculos	✓			✓		20 ns
Acceso mem. lectura	✓		✓			30 ns
Acceso mem. escritura	✓				✓	35 ns
Escritura registro	✓	✓				15 ns

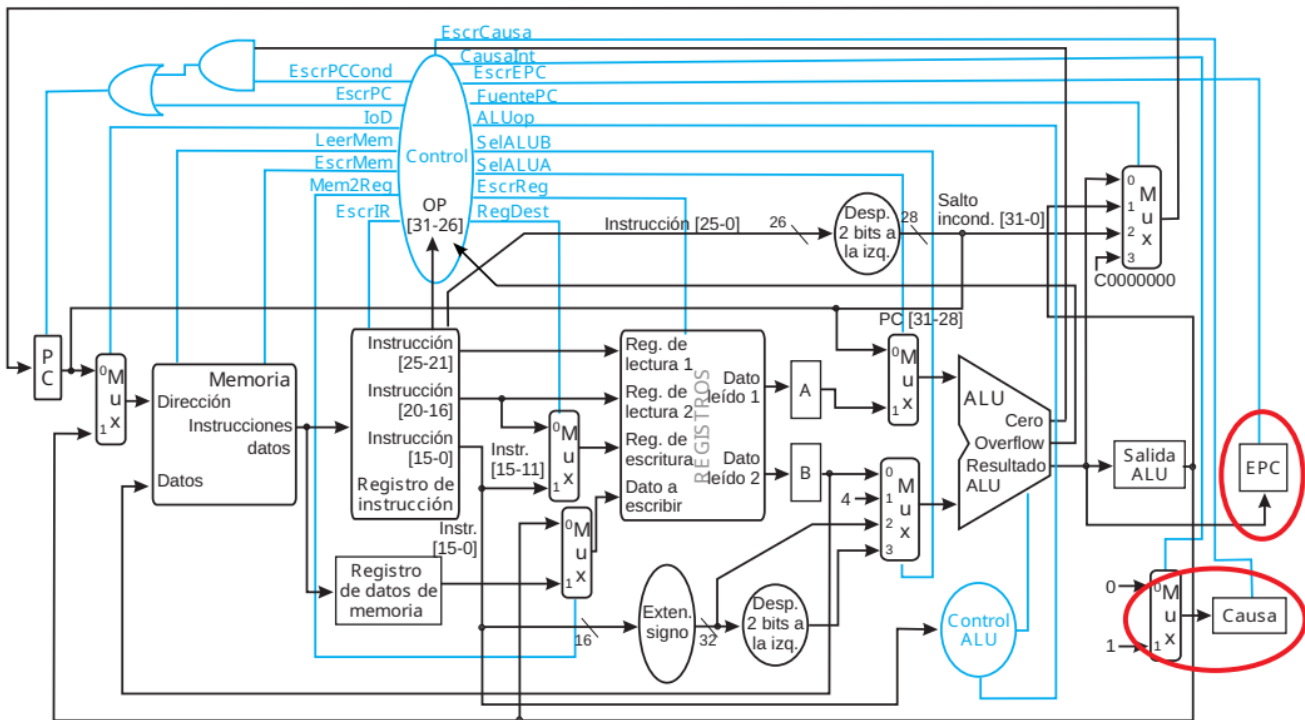
- El ciclo del reloj debería ser al menos tan largo como la más larga de las operaciones del camino de datos (Ejemplo Ciclo de reloj $\geq 35\text{ns}$)
 - lw: 175 ns (cinco ciclos)
 - sw e instrucciones A-L: 140ns (cuatro ciclos)
 - beq u j: 105 ns (tres ciclos)

Procesamiento de excepciones

Excepciones

- Excepción: Suceso inesperado ocurrido en el procesador.
- Acciones a llevar a cabo ante una excepción:
 - Guardar la dirección de la instrucción causante en el registro contador de programa de excepción (EPC.)
 - Transferir el control al SO, que ejecutará alguna ruta específica.
 - Finalizar el programa o continuar su ejecución (utilizando EPC).
- Tipos de excepciones en nuestra implementación de MIPS:
 - Ejecución de una instrucción no definida.
 - Desbordamiento aritmético.
- La rutina del SO a ejecutar dependerá del tipo de excepción.
- En el MIPS se incluye un registro denominado Registro de Causa (Causa) para almacenar la razón de la excepción
 - Causa = 1: instrucción no definida
 - Causa = 0: desbordamiento aritmético
- EPC: registro de dirección de la instrucción causante (32 bits).
- Causa: Registro para almacenar la causa de la excepción (1 bit).
 - Causa = 1: Instrucción no definida.
 - Causa = 0: Desbordamiento aritmético.

Camino de datos modificado

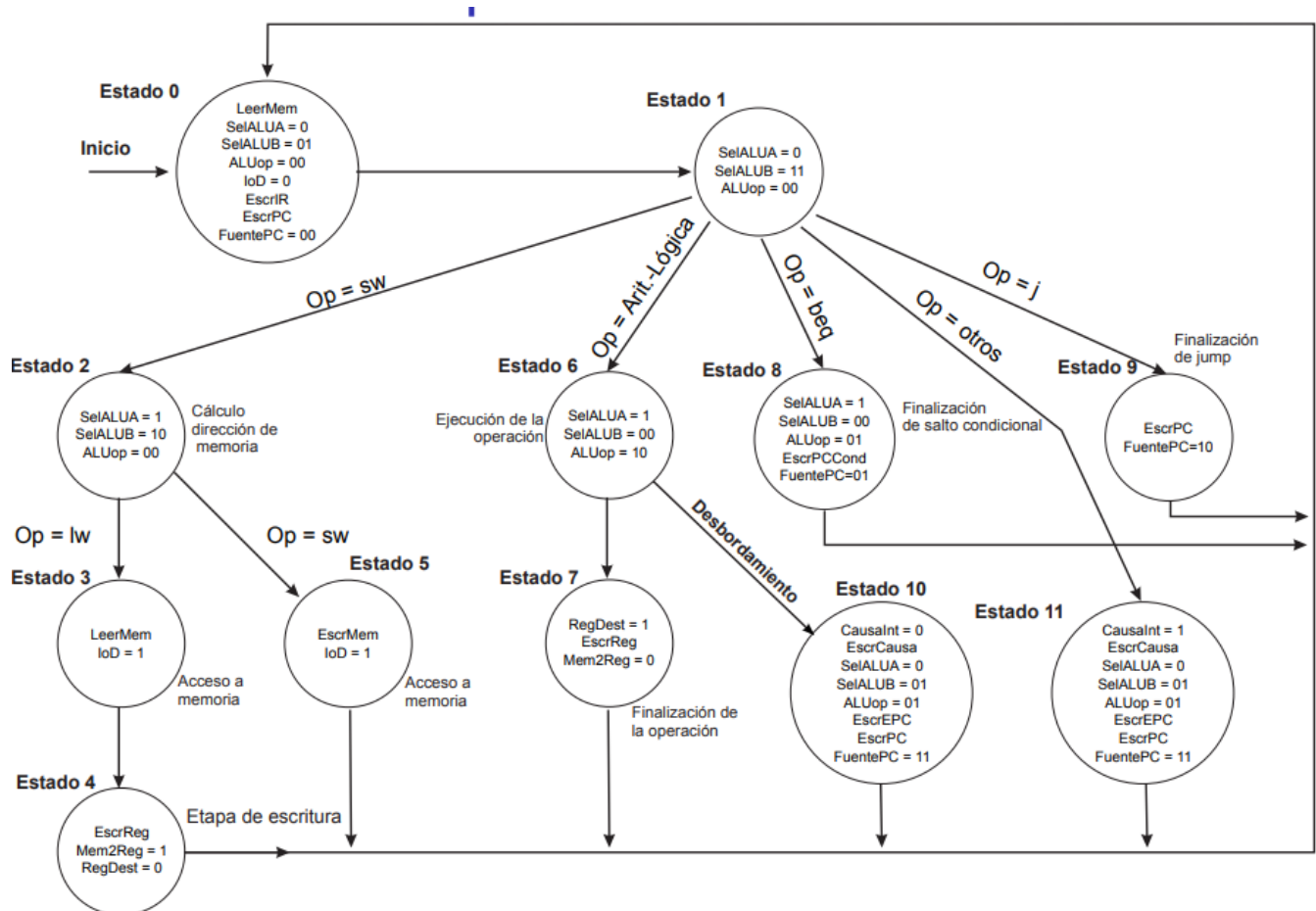


- Señales de control:
 - EscrEPC.
 - EscrCausa
 - CausalInt: valor del registro Causa (1 bit).
- Nueva entrada al MUX del PC: 0xC0000000 (entrada al código del SO)

Procesamiento de una excepción

- Escribir en el registro Causa un 0 o un 1.
- Guardar la dirección de la instrucción causante (PC-4) en EPC.
- Poner en el PC la dirección de procesamiento de excepción del SO.

Unidad de control completa



- Instrucción no definida:
 - Se detecta al no estar definido el estado siguiente al estado 1 para el código de operación encontrado.
 - El estado siguiente será el estado 11.
- Desbordamiento aritmético:
 - Se detecta cuando la ejecución de una instrucción aritmético-lógica (estado 6) arroja un desbordamiento
 - El estado siguiente será el estado 10.

Microprogramación

- Modificaciones a la UC microprogramada:
 - 2 nuevas microinstrucciones (estados 10 y 11)
 - Modifican tabla envío 1
 - Nueva tabla de envío 3 para bifurcación estado 6
 - Microinstrucciones de tamaño 22 bits en vez de 18:
 - CrtlDir podrá tomar 5 valores diferentes: ampliación a 3 bits
 - Nuevas señales de control: CausalInt, EscrCausa, EscrEPC

Código de operación	instrucción	valor
000000	Tipo R	Rformat1
000001	jmp	JUMP1
000100	beq	BEQ1

Código de operación	instrucción	valor
100011	lw	Mem1
101011	sw	Mem 1
otro	otra	Nodef1

Señal de desobrdamiento	Valor
0	Arti3
1	Ov3

Mejoras del rendimiento en los procesadores actuales

- Hay dos posibilidades para incrementar el rendimiento:
 - Mejor la tecnología de los circuitos integrados, lo cual reduciría el tiempo necesario para ejecutar cada una de las etapas y permitiría reducir el ciclo de reloj.
 - Mejorar el diseño del procesador:
 - Procesador Segmentado: Trabaja con varias instrucciones a la vez, cada una en una etapa diferente. Se pretende conseguir la ejecución de una instrucción por ciclo (límite máximo).
 - Procesador Superscalar: Implementa paralelismo a nivel de instruction. Ejecuta más de 1 instrucción por ciclo de reloj gracias a la replicación de unidades funcionales.
- Todas las CPUs de propósito general actuales son segmentadas y superescalares.