Diseño de la Ruta de Datos y la Unidad de Control

Introducción

Contenidos

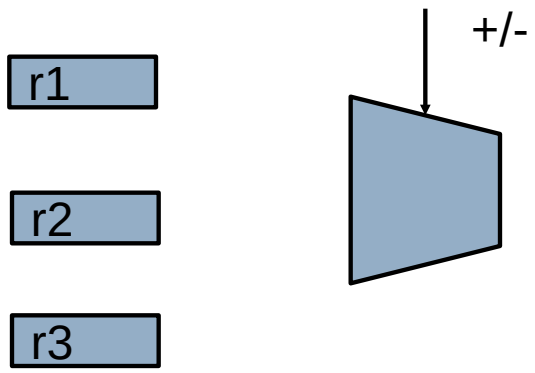
- Introducción
  - Importancia del diseño del procesador. Metodología de diseño de un procesador. Arquitectura MIPS: formato de la instrucción máquina y repertorio de instrucciones.
- Diseño de la ruta de datos monociclo
  - Componentes de la ruta de datos. Ensamblaje de la ruta de datos. Ruta de datos monociclo: puntos de control.
- Diseño del controlador monociclo
  - Determinación de los valores de los puntos de control. Control global vs. Control local. Ruta datos monociclo + controlador. Temporización monociclo.
- Diseño de la ruta de datos (multiciclo)
  - Ruta de datos multiciclo: con y sin buses.
- Diseño del controlador (multiciclo)
  - Diagrama de estados del controlador. El controlador como una FSM. Alternativas de implementación del controlador

Introducción

Metodología para el diseño de un procesador

- **Paso 1** **Analizar el repertorio de instrucciones** para obtener los requisitos de la ruta de datos
  - **Ruta de datos**
    - **elementos de almacenamiento**
      - **visibles / transparentes** en el nivel de ISA (programador / compilador)
    - **Unidades funcionales**
    - **Elementos de direccionamiento e interconexión**
  - Instrucción
    - Conjunto de transferencias entre registros.
    - La ruta de datos debe ser capaz de soportar dichas transferencias.

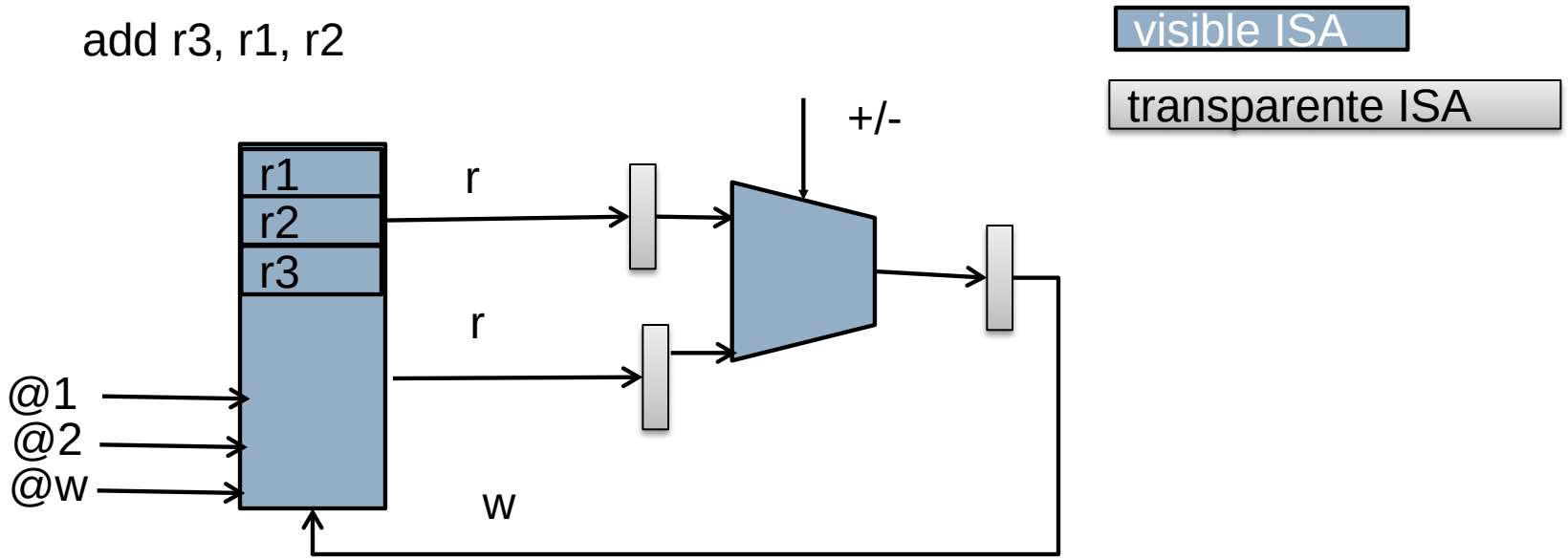
add r3, r1, r2



Introducción

Metodología para el diseño de un procesador

- **Paso 2 Establecer la metodología de temporización**
  - **Monociclo (CPI = 1)**: La instrucción se ejecuta en un ciclo
  - **Multiciclo (CPI > 1)**: La instrucción se ejecuta en varios ciclos
- **Paso 3 Seleccionar el conjunto de módulos** (de almacenamiento, operativos e interconexión) que forman la ruta de datos

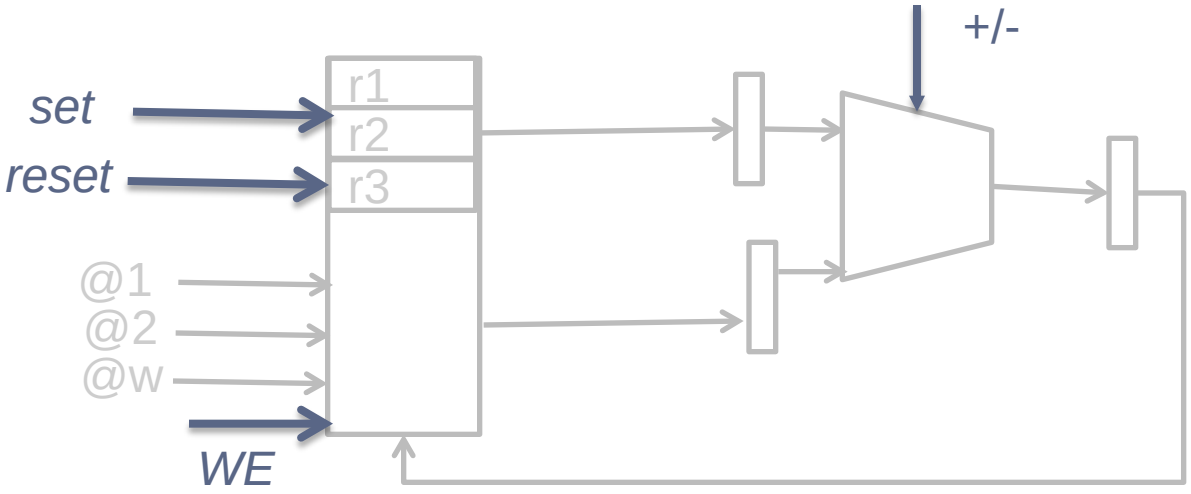


Introducción

Metodología para el diseño de un procesador

- **Paso 4** **Ensamblar la ruta de datos** de modo que se cumplan los requisitos impuestos por el repertorio, **localizando los puntos de control**
- **Paso 5** **Determinar los valores de los puntos de control** analizando las transferencias entre registros incluidas en cada instrucción.
- **Paso 6** **Diseñar la lógica de control.**

add r3, r1, 32



Introducción
Arquitectura MIPS

- **Procesadores RISC de 32 y 64 bits**
- **Repertorio ortogonal**
  - **Aprox. 100 instrucc. máquina + 30 instrucc. de aritmética en punto flotante**
    - **Un tamaño de operandos y un modo de direccionamiento por instrucción**
  - **Instrucciones: cuatro grupos**
    - **Movimiento de datos**
    - **Aritmética entera, lógicas y desplazamiento**
    - **Control de flujo**
    - **Aritmética en punto flotante**
  - **4 modos de direccionamiento**
    - **Inmediato**
    - **Directo de registros**
    - **Indirecto con desplazamiento**
    - **Indirecto con desplazamiento relativo al PC**
  - **3 formatos de instrucción distintos con longitud única de 32 bits**

Introducción
Arquitectura MIPS

- Arquitectura registro-registro
  - Sólo LOAD y STORE para referencia a memoria
  - Resto de instrucciones: operan sobre registros
  - Instrucciones con tres operandos
    - 2 op. fuente y 1 op. Destino
    - Notación ensamblador:
      - **`op x, y, z`** ; *x <- (y) op (z)*

Introducción
Arquitectura MIPS

- Banco de 64 registros (32 bits cada uno)
  - 32 de propósito general (R0-R31)
  - 32 para inst. en punto flotante (F0-F31).
    - Pueden usarse como:
      - 32 registros para ops en simple precisión (32 bits)
      - 16 registros para ops en doble precisión (64 bit)

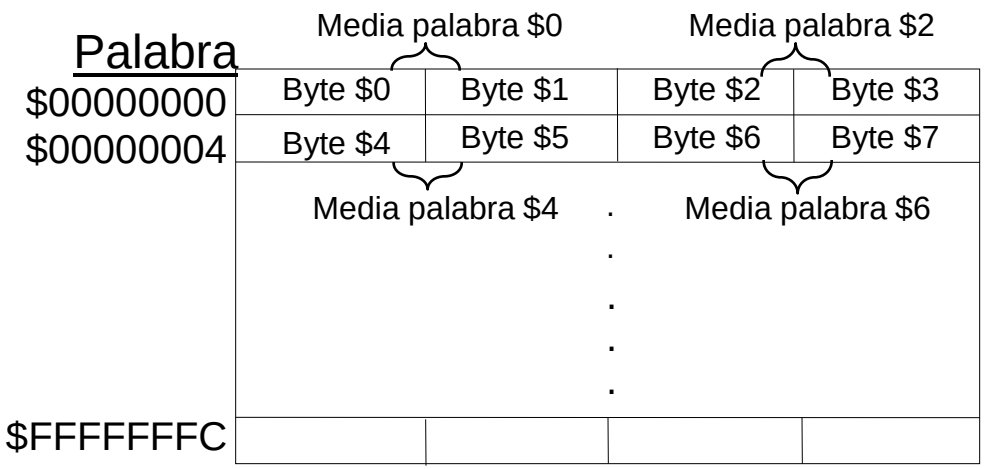
Introducción

Tipos de datos arquitectura MIPS

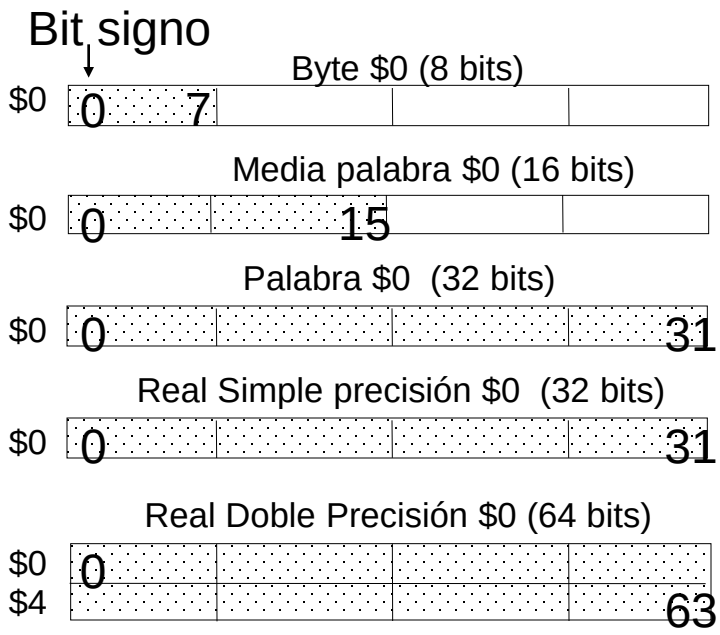
- Enteros
  - **Tamaño** Byte **(8 bits)**
  - **Tamaño** Media palabra **(16 bits)**
  - **Tamaño** Palabra **(32 bits)**

- Reales en punto flotante
  - Simple precisión **(32 bits)**
  - Doble precisión **(64 bits)**

Memoria (Ordenación Little-Endian)



Nota: Soporta tanto big-endian como little-endian



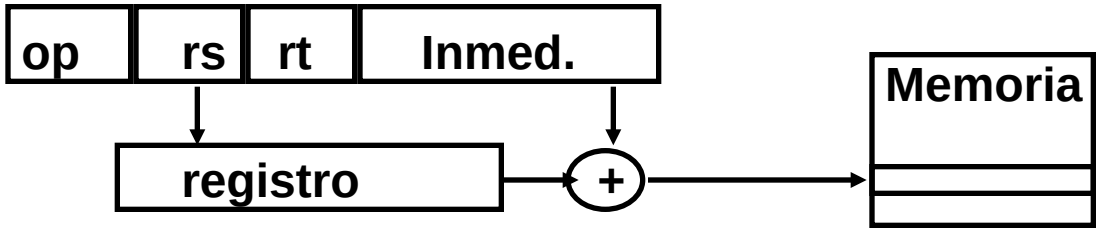
Introducción

Modos de direccionamiento MIPS

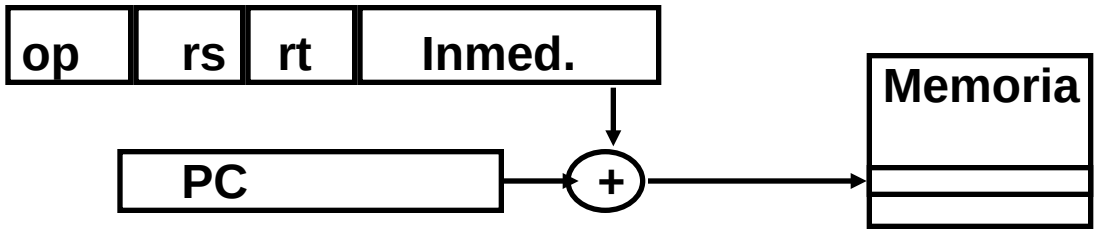
- **Directo de registro**

- **Inmediato**

- **Indirecto con desplazamiento**



- **Indirecto con desplazamiento relativo a PC**



Introducción

Arquitectura MIPS: Instrucciones que vamos a implementar

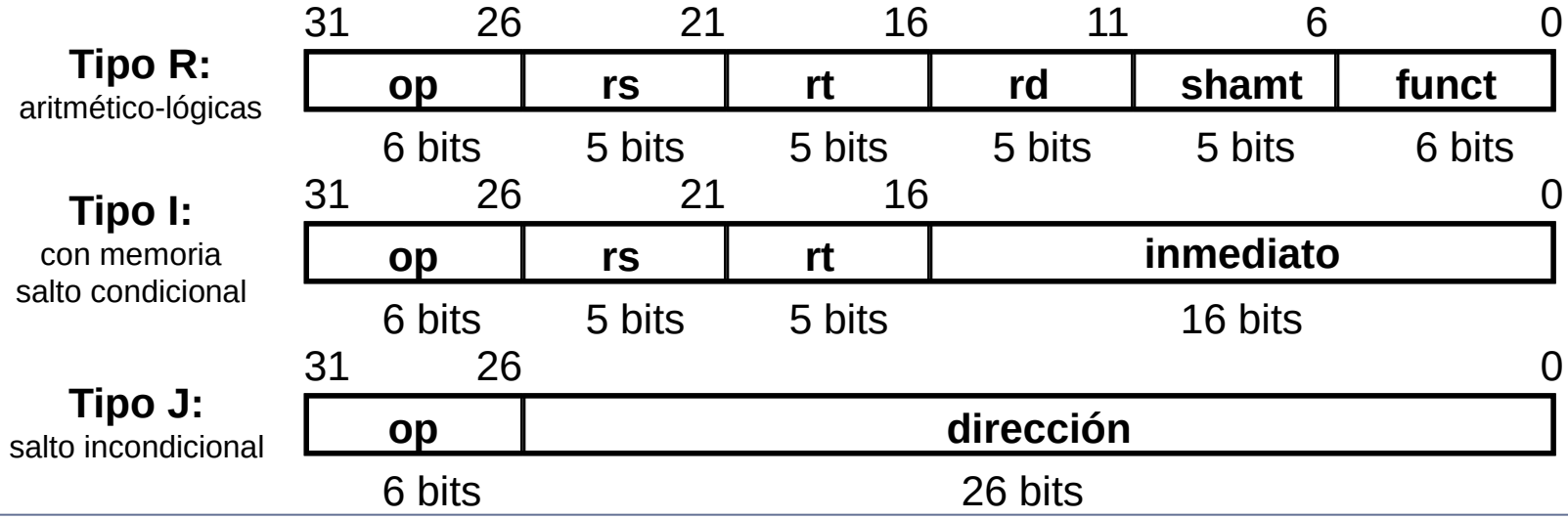
- **Instrucciones con referencia a memoria** (formato tipo I):
  - lw rt, inmed(rs) rt ← Memoria(rs + SignExt(inmed)) , PC ← PC + 4
  - sw rt, inmed(rs) Memoria(rs + SignExt(inmed)) ← rt , PC ← PC + 4
- **Instrucciones aritmético-lógicas con operandos en registros** (formato tipo R)
  - add rd, rs, rt rd ← rs + rt, PC ← PC + 4
  - sub rd, rs, rt rd ← rs - rt , PC ← PC + 4
  - and rd, rs, rt rd ← rs and rt , PC ← PC + 4
  - or rd, rs, rt rd ← rs or rt , PC ← PC + 4
- **Instrucciones de salto condicional** (formato tipo I)
  - beq rs, rt, inmed si (rs = rt) entonces (PC ← PC + 4 + 4·SignExp(inmed))
    en otro caso PC ← PC + 4

1. El ciclo de instrucción comienza buscando la instrucción en memoria (*fetch*)
   - instrucción ← Memoria(PC)
2. En función del tipo de instrucción se realiza una de las anteriores operaciones
3. Se vuelve a comenzar

Introducción

Arquitectura MIPS: formato de la instrucción máquina

- Todas las instrucciones del repertorio del MIPS tienen 32 bits de anchura, repartidas en 3 formatos de instrucción diferentes:

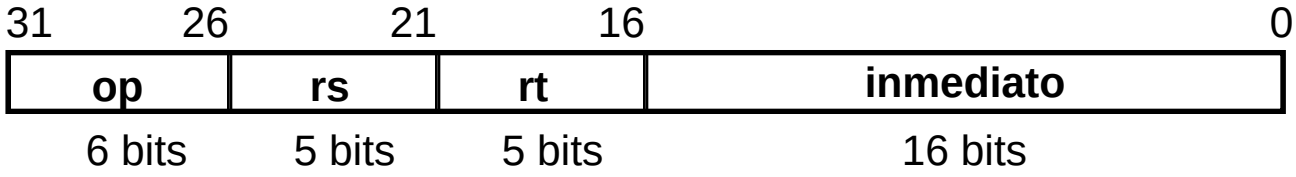


- El significado de los campos es:
  - **op**: identificador de instrucción
  - **rs**, **rt**, **rd**: identificadores de los registros fuentes y destino
  - **shamt**: cantidad a desplazar (en operaciones de desplazamiento)
  - **funct**: selecciona la operación aritmética a realizar
  - **inmediato**: operando inmediato o desplazamiento en direccionamiento a registro-base
  - **dirección**: dirección destino del salto

Introducción

Arquitectura MIPS: formato de la instrucción máquina

Tipo I:
con memoria
salto condicional



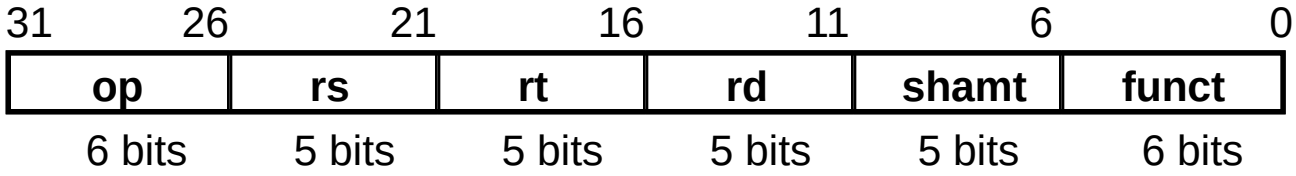
- **Instrucciones con referencia a memoria** (formato tipo I):
  - lw rt, inmed(rs) **rt** ← Memoria(**rs** + SignExt(**inmed**)) , PC ← PC + 4
  - sw rt, inmed(rs) Memoria(**rs** + SignExt(**inmed**)) ← **rt** , PC ← PC + 4

- **Operaciones a realizar:**
  1. **Leer la instrucción de la memoria de instrucciones**
  2. **Decodificarla para identificar el tipo de instrucción y sus operandos**
  3. **Leer los operandos de los registros**
  4. Extender el signo del operando inmediato
  5. Realizar la suma de **rs** con el operando inmediato
  6. Acceder a memoria para leer o escribir
  7. En caso de la instrucción **lw** se escribe el dato leído de memoria en el registro **rt**.
  8. **Sumar 4 al contador de programa**

Introducción

Arquitectura MIPS: formato de la instrucción máquina

Tipo R:
aritmético-lógicas



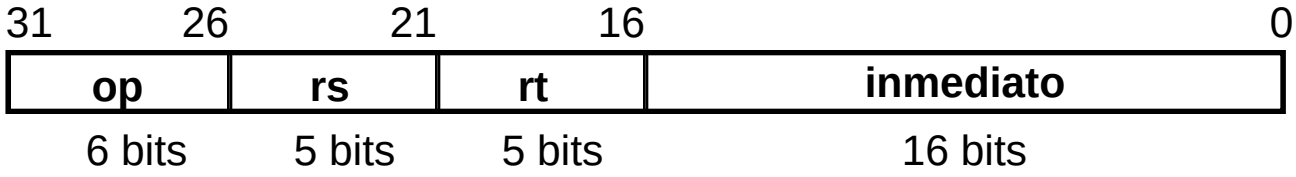
- **Instrucciones aritmético-lógicas con operandos en registros** (formato tipo R)
  - add rd, rs, rt — rd ← rs + rt, PC ← PC + 4
  - sub rd, rs, rt — rd ← rs - rt , PC ← PC + 4
  - and rd, rs, rt — rd ← rs and rt , PC ← PC + 4
  - or rd, rs, rt — rd ← rs or rt , PC ← PC + 4

- **Operaciones a realizar:**
  1. **Leer la instrucción de la memoria de instrucciones**
  2. **Decodificarla para identificar el tipo de instrucción y sus operandos**
  3. **Leer los operandos de los registros (rs y rt)**
  4. Realizar la operación indicada en **funct**
  5. Escribir el resultado en el registro **rd**.
  6. **Sumar 4 al contador de programa**

Introducción

Arquitectura MIPS: formato de la instrucción máquina

Tipo J:
con memoria
salto condicional



- **Instrucciones de salto condicional** (formato tipo I)
  - beq rs, rt, inmed si (rs = rt) entonces (PC ← PC + 4 + 4·SignExp(inmed))
    en otro caso PC ← PC + 4

- **Operaciones a realizar:**
  1. **Leer la instrucción de la memoria de instrucciones**
  2. **Decodificarla para identificar el tipo de instrucción y sus operandos**
  3. **Leer los operandos de los registros**
  4. Comparar rs con rt (se realiza una resta y se comprueba si el resultado es cero)
  5. Extender el signo del operando inmediato
  6. **Sumar 4 al contador de programa**
  7. Si se cumple la condición sumar el desplazamiento indicado en el operando inmediato