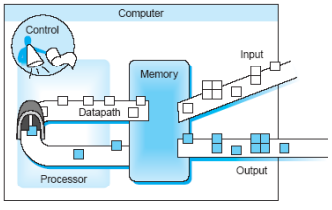
El procesador: Datapath y la Unidad de Control

Arquitectura de Computadoras
Primavera 2012

Contenido

Introducción

- Los elementos básicos de un sistema de computo son:



Introducción

- Nos centraremos en diseñar una implementación que incluya un subconjuto del conjunto de instrucciones discutido:
  - Instrucciones de referencia a memoria lw y sw
  - Instrucciones aritméticas y lógicas add, sub, and, or y slt
  - Instrucciones de decisión beq y salto incondicional j
- Solo incluimos este subconjunto para hacerlo mas claro e ilustrar los puntos claves de diseño

Introducción

- Todas las instrucciones del conjunto discutido requieren de los siguientes dos pasos:
  - Obtener el código de la instrucción de la memoria
    - El contador del programa (PC) tiene la dirección de memoria donde se encuentra el código
    - Se utiliza PC para ir a memoria y obtener el código
  - Leer uno o dos registros
    - Se utilizan los campos de la instrucción para seleccionar los registros que se leerán
    - En algunos casos (Instrucciones tipo I) solo leemos un registro y utilizamos otro campo de la instrucción como operando

Introducción

- Después de estos dos pasos, la ejecución de la instrucción requerirá acciones diferentes.
- Sin embargo, muchas de estas instrucciones requieren pasos similares.
  - Por ejemplo:
    - add obtiene los dos registros y usa ALU para sumarlos
    - lw obtiene un registro y lo suma a uno de sus campos por medio de la ALU para obtener la dirección de memoria que debe leer

Construyendo el datapath

1. Analizar el conjunto de instrucciones => requerimientos del datapath
2. Seleccionar el conjunto de componentes del datapath y establecer la metodología de sincronización
3. Construir el datapath que satisfaga los requerimientos
4. Analizar la implementación de cada instrucción para determinar los puntos de control
5. Diseñar la unidad de control lógico

7

Analizar el conjunto de instrucciones

- Todas las instrucciones siguen uno de los siguientes formatos

Formato R

OPCODE	RS	RT	RD	SHAMT	FUNCTION
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Formato I

OPCODE	RS	RT	Dirección/Numero Inmediato
6 bits	5 bits	5 bits	16 bits

Formato J

OPCODE	Dirección
6-bits	26-bits

8

El subconjunto de Instrucciones

Instrucción	Transferencia de Registro	PC
add	R[rd] <- R[rs] + R[rt];	PC <- PC + 4
sub	R[rd] <- R[rs] - R[rt];	PC <- PC + 4
and	R[rd] <- R[rs] and R[rt];	PC <- PC + 4
ori	R[rt] <- R[rs] or zero_ext(Imm16);	PC <- PC + 4
slt	R[rd] <- 1 if R[rs] < R[rt] else 0	PC <- PC + 4
lw	R[rt] <- MEM[R[rs] + sign_ext(Imm16)]	PC <- PC + 4
sw	R[rt] -> MEM[R[rs] + sign_ext(Imm16)]	PC <- PC + 4
beq	IF (R[rs] == R[rt]) Then	PC <- PC + sign_ext(Imm16<<2)]
	else	PC <- PC + 4
j		PC <- PC:(Target<<2)

9

Paso 1: Requerimiento del Conjunto de Instrucciones

- Memoria
  - Las instrucciones deben estar almacenadas en la memoria para poder ser ejecutadas
  - Algunas instrucciones (lw, sw) requieren leer o guardar datos en memoria
- Registros
  - La mayor parte de las instrucciones requiere acceder a información guardada en los registros (rt, rs)
  - Algunas instrucciones deben guardar los resultados dentro de un registro (rt o rd)

10

Paso 1: Requerimiento del Conjunto de Instrucciones

- Registro de Control de Programa (PC)
  - Este registro nos permitirá seguir la secuencia de instrucciones del programa
- Extensor
  - Algunas operaciones se realizan con números de 16 bits. Por lo cual, se debe extender el número a 32-bits para poder realizar la operación
- Sumador para PC
  - Se requiere sumar un 4 a PC o un número extendido a 16 bits a PC

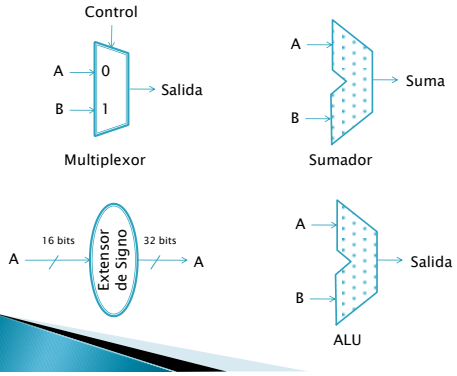
11

Paso 2: Componentes del Datapath

- Elementos Combinacionales
  - ALU
  - Sumador
  - Extensor de signo
  - Multiplexores
- Elementos de almacenamiento
  - Memoria
  - Registros
    - Metodología de sincronización

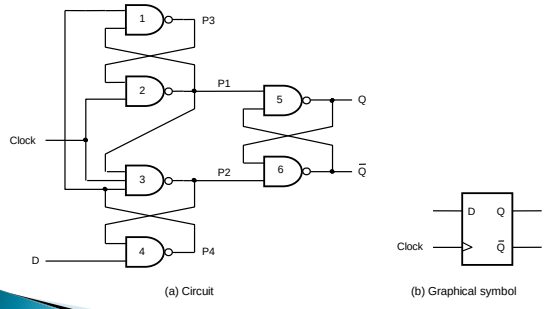
12

Elementos Combinacionales



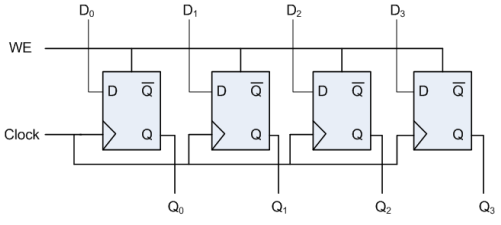
13

Flip-Flop tipo D activado con el flanco positivo del reloj



14

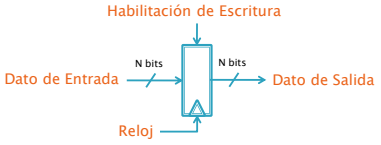
Registro de 4-bits con habilitacion de escritura



15

Elementos de Almacenamiento

- Registros
  - Los registros son una colección de N flip-flops tipo D con habilitación de escritura

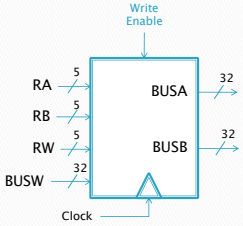


  - Cuando la habilitación de escritura es 0, el dato de salida no cambia
  - Cuando la habilitación de escritura es 1, el dato de salida es el dato de entrada

16

Elementos de Almacenamiento

- Archivo de Registros
  - El archivo de registro contiene 32 registros de 32 bits
  - Tiene 2 buses de lectura BUSA y BUSB cada uno de 32 bits
  - Cuenta con 1 bus de escritura BUSW
  - El registro se selecciona por medio de:
    - RA: selecciona el registro de lectura A
    - RB: selecciona el registro de lectura B
    - RW: selecciona el registro de escritura W
  - El reloj solo es un factor en la escritura

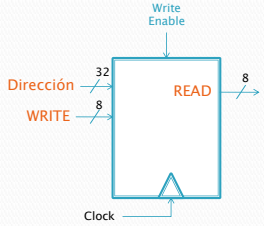


17

Elementos de Almacenamiento

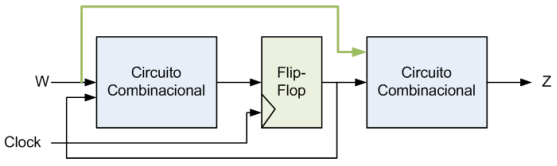
- Memoria (Idealizada)
  - Un puerto de entrada WRITE
  - Un puerto de salida READ
- Una palabra de memoria se selecciona por medio de:
  - Dirección
  - WriteEnable = 1, entonces en la palabra seleccionada se escribe lo que se encuentre en WRITE
- Reloj

Memoria

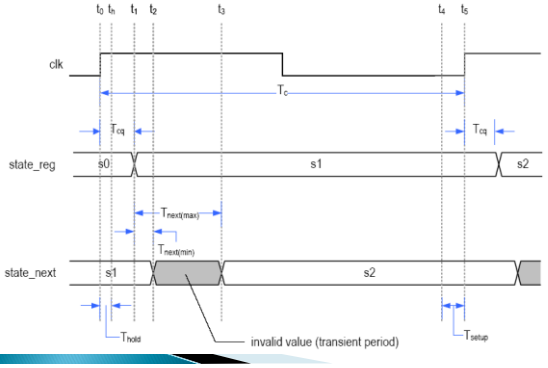


18

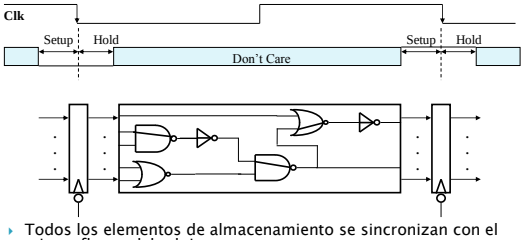
Circuito Secuencial Basico



Sincronización



Metodología de Sincronización



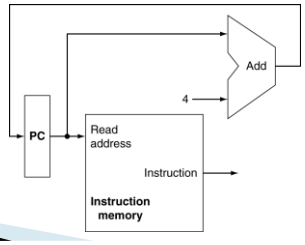
- Todos los elementos de almacenamiento se sincronizan con el mismo flanco del reloj
- Cycle Time = CLK-to-Q + Longest Delay Path + Setup + Clock Skew
- (CLK-to-Q + Shortest Delay Path – Clock Skew) > Hold Time

PASO 3: Construir el datapath

- Requerimientos de transferencia de registros
  - Ensamblado del datapath
- Obtención de la instrucción
- Lectura de operandos
- Ejecución de instrucción

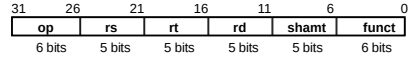
Ensamblando el datapath

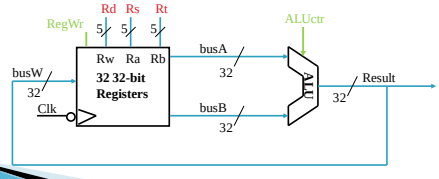
- Obteniendo la instrucción MEM[PC]
- Actualizando el contador del programa
  - PC <- PC + 4
  - PC <- PC + "Algo mas" para beq y j



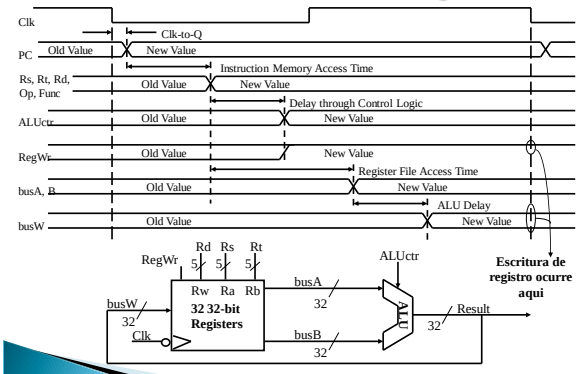
Suma y Resta

- R[rd] <- R[rs] op R[rt], Ejemplo: add rd, rs, rt
  - Ra, Rb, y Rw provienen de los campos de la instrucción rs, rt, y rd
  - ALUctr y RegWr: lógica de control después de decodificar la instrucción



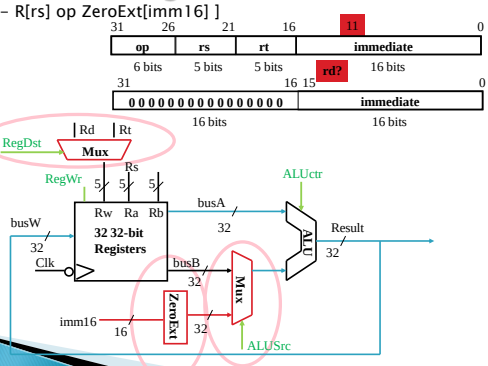


Transferencia hacia el archivo de registros



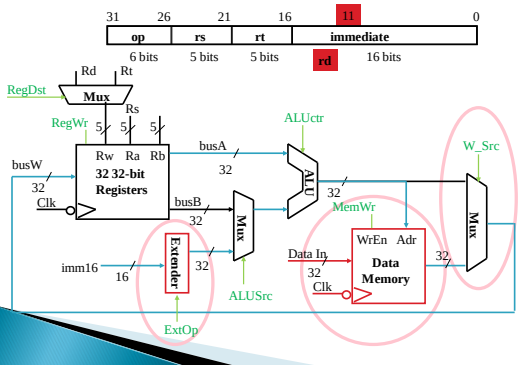
Operaciones lógicas con inmediatos

R[rt] <- R[rs] op ZeroExt[imm16]]



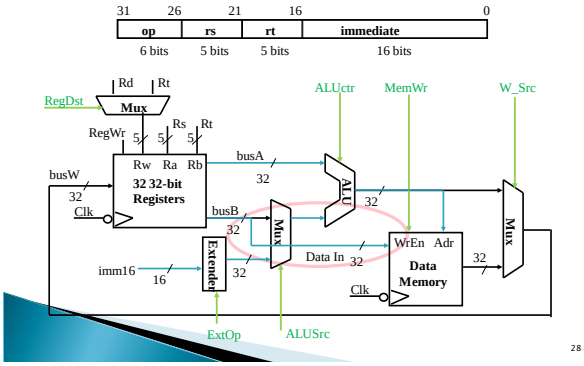
3d: Operación de lectura de Memoria lw

- R[rt] <- Mem[R[rs] + SignExt[imm16]] ejemplo: lw rt, rs, imm16

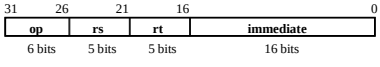


3e: Operación de Almacenamiento de Memoria

- Mem[R[rs] + SignExt[imm16] <- R[rt]] Ejemplo: sw rt, rs, imm16



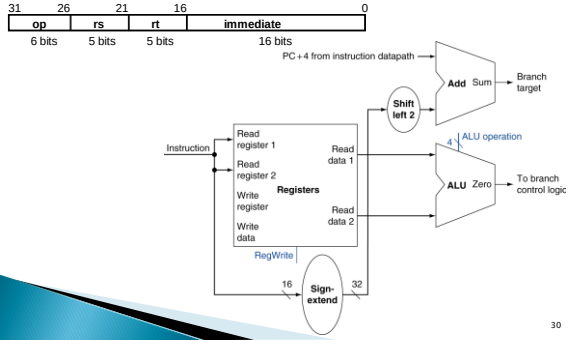
3f: Instrucción de toma de decisiones



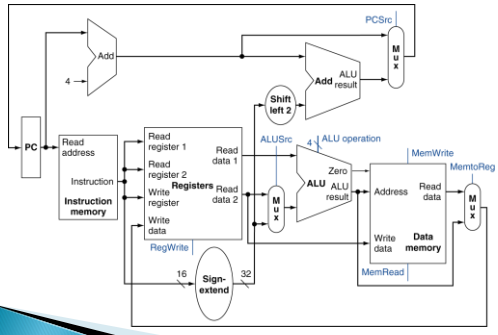
- beq rs, rt, imm16
  - mem[PC] Obten la instr. de memoria
  - Igual <- R[rs] == R[rt] Calcula la condición de salto
  - if (COND eq 0) Calcula la dirección de la siguiente instrucción
    - PC <- PC + 4 + (SignExt(imm16) x 4)
  - else
    - PC <- PC + 4

Datapath para saltos condicionales

- beq rs, rt, imm16 Datapath evalua la condición



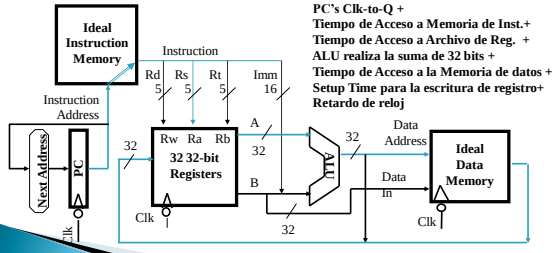
Datapath Completo



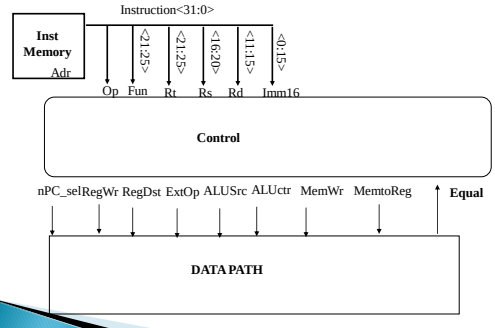
Vista Abstracta del Camino Critico

- El archivo de registros y la Memoria Ideal:
  - El reloj es un factor solamente durante la operación de escritura
  - Durante la operación de lectura, se comporta como lógica combinacional:
    - Dirección valida=> Salida después de "Tiempo de acceso."

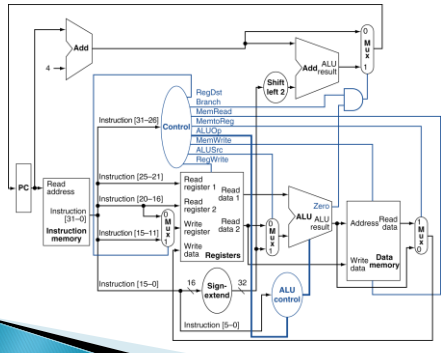
Critical Path (Operacion LW) =
PC's Clk-to-Q +
Tiempo de Acceso a Memoria de Inst.+
Tiempo de Acceso a Archivo de Reg. +
ALU realiza la suma de 32 bits +
Tiempo de Acceso a la Memoria de datos +
Setup Time para la escritura de registro+
Retardo de reloj



Paso 4: Unidad de Control



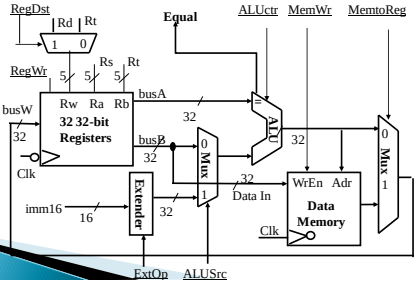
Datapath With Control



Señales de Control

- ExtOp: "zero", "sign"
- ALUsrc:
  - 0 => regB;
  - 1 => immed
- ALUctr: "add", "sub", "or"

- MemWr: Escritura en memoria
- MemtoReg: 1 => Mem
- RegDst: 0 => "rt"; 1 => "rd"
- RegWr: Escritura a reg. destino



Señales de Control

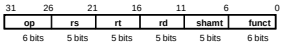
inst	Register Transfer	
ADD	R[rd] <– R[rs] + R[rt];	PC <– PC + 4
ALUsrc = RegB, ALUctr = "add", RegDst = rd, RegWr, nPC_sel = "+4"		
SUB	R[rd] <– R[rs] – R[rt];	PC <– PC + 4
ALUsrc = ___, Extop = __, ALUctr = ___, RegDst = ___, RegWr(?), MemtoReg(?), MemWr(?), nPC_sel =__		
ORi	R[rt] <– R[rs] + zero_ext(Imm16);	PC <– PC + 4
ALUsrc = ___, Extop = __, ALUctr = ___, RegDst = ___, RegWr(?), MemtoReg(?), MemWr(?), nPC_sel =__		
LOAD	R[rt] <– MEM[R[rs] + sign_ext(Imm16)];	PC <– PC + 4
ALUsrc = ___, Extop = __, ALUctr = ___, RegDst = ___, RegWr(?), MemtoReg(?), MemWr(?), nPC_sel =__		
STORE	MEM[R[rs] + sign_ext(Imm16)] <– R[rs];	PC <– PC + 4
ALUsrc = ___, Extop = __, ALUctr = ___, RegDst = ___, RegWr(?), MemtoReg(?), MemWr(?), nPC_sel =__		
BEQ	if (R[rs] == R[rt]) then PC <– PC + sign_ext(Imm16)] || 00 else PC <– PC + 4	
ALUsrc = ___, Extop = __, ALUctr = ___, RegDst = ___, RegWr(?), MemtoReg(?), MemWr(?), nPC_sel =__		

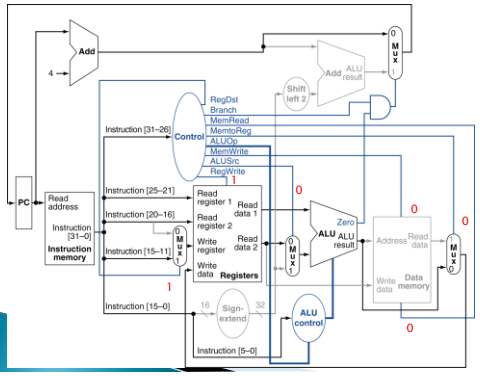
Paso 5: Lógica para cada señal de control

- nPC_sel <= if (OP == BEQ) then EQUAL else 0
- ALUsrc <= if (OP == "000000") then "regB" else "immed"
- ALUctr <= if (OP == "000000") then funct
  elseif (OP == ORi) then "OR"
  elseif (OP == BEQ) then "sub"
  else "add"
- ExtOp <= _____
- MemWr <= _____
- MemtoReg <= _____
- RegWr: <=_____
- RegDst: <= _____

38

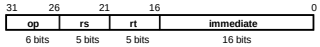
R-Type

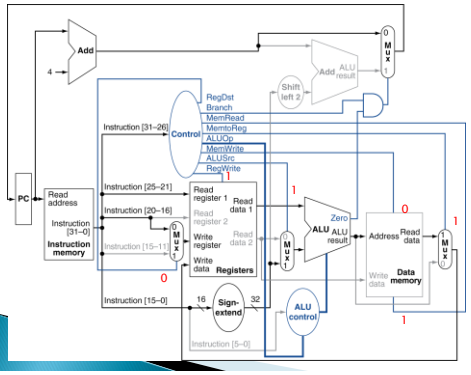




40

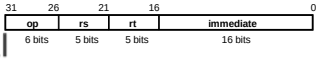
Load

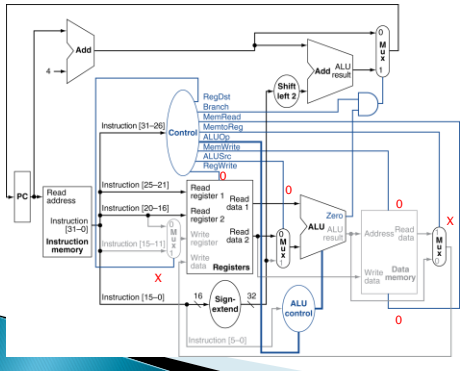




41

Branch-on-Equal

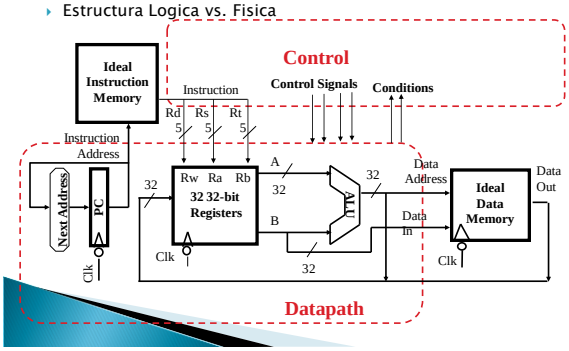




42

Vista Abstracta de la Implementación

- Estructura Logica vs. Fisica



43