



La mayor parte de las figuras corresponden al libro de Andrew Tanenbaum, Organización de Computadoras (un enfoque estructurado), 4ta. edición



Ejecución de instrucciones

- Para poder ejecutar las instrucciones la CPU posee registros internos
 - Contador de Programa (PC) o Registro de Próxima Instrucción (RPI)
 - Registro de Instrucción (IR)
 - Registros de Uso General ($R_1 \dots R_n$)
 - Registro de Direcciones de Memoria (RDM o MAR)
 - Registro de Datos de Memoria (RBM o MDR)
 - Otros

Arquitectura de los Sistemas de Cómputo

1



Ejecución de instrucciones (cont.)

- La ejecución de una instrucción (ciclo de instrucción) puede dividirse en dos semiciclos
 - Búsqueda de la instrucción (incluye la actualización del PC)
 - Ejecución de la instrucción
- Cada uno de los semiciclos puede, a su vez, descomponerse en instrucciones aún más elementales (microinstrucciones)
- Estas microinstrucciones son, básicamente:
 - Transferencias entre registros
 - Ordenes a los restantes dispositivos (ALU, memoria, etc.)

Arquitectura de los Sistemas de Cómputo

2



Ejemplos

- Buscar una palabra en memoria
 - $MAR \leftarrow (R_1)$
 - LEER
 - ESPERAR
 - $R_2 \leftarrow (MDR)$
- Almacenamiento de una palabra en memoria
 - $MAR \leftarrow (R_1)$
 - $MDR \leftarrow (R_2)$
 - ESCRIBIR
 - ESPERAR

Arquitectura de los Sistemas de Cómputo

3



Transferencia entre registros

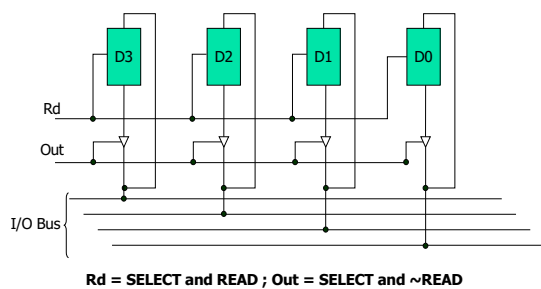
- Las operaciones de transferencia entre registros dependen de la estructura interna de la CPU
- Un registro puede estar conectado a un único bus (de entrada salida), a dos buses (uno de entrada y otro de salida), a tres buses, etc.
- En general, a mayor número de buses mayor la posibilidad de ejecutar en forma más eficiente las operaciones

Arquitectura de los Sistemas de Cómputo

4

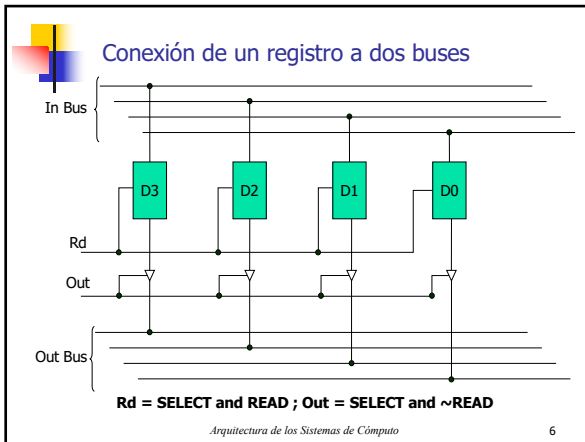


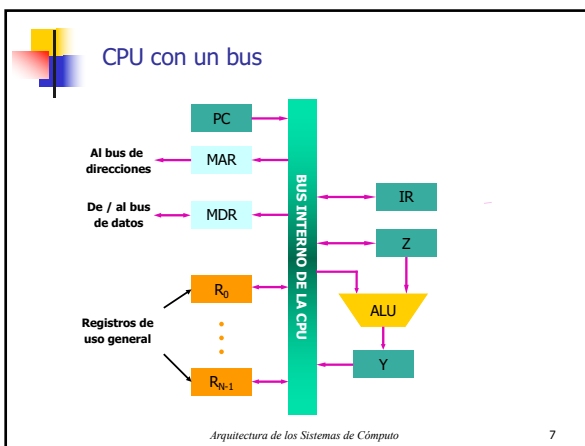
Conexión de un registro a un bus

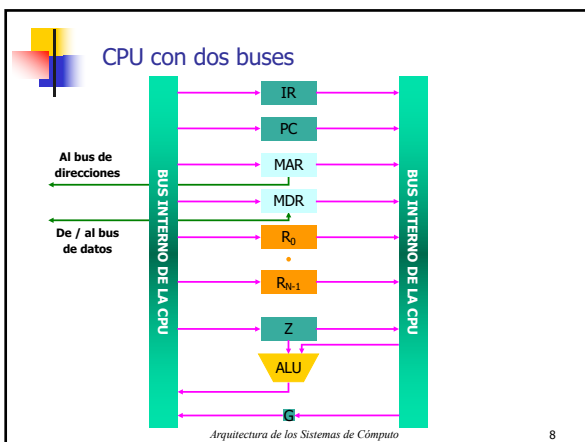


Arquitectura de los Sistemas de Cómputo

5









Ejemplos de microinstrucciones

- Transferencia entre registros
 - $R_1 \text{ in}, R_2 \text{ out} \sim R_1 \leftarrow (R_2)$
- Operación aritmético lógica (con un bus)
 - $Z \text{ in}, R_1 \text{ out} \sim Z \leftarrow (R_1)$
 - $R_2 \text{ out}, \text{ADD}, Y \text{ in} \sim Y \leftarrow (Z) + (R_2)$
 - $Y \text{ out}, R_3 \text{ in} \sim R_3 \leftarrow Y$
- La misma operación (con dos buses)
 - $R_1 \text{ out}, G \text{ hab}, Z \text{ in} \sim Z \leftarrow (R_1)$
 - $R_2 \text{ out}, \text{ADD}, \text{ALU out}, R_3 \text{ in} \sim R_3 \leftarrow (Z) + (R_2)$

Arquitectura de los Sistemas de Cómputo

9



Ejecución completa de una instrucción

- $R_1 \leftarrow (R_1) + (\text{Mem IR}_{\text{OPE}})$
 - Buscar la instrucción y actualizar el PC
 1. PC out, MAR in, LEER, CLEAR Z, CARRY = 1; ADD, Y in
 2. Y out, PC in, ESPERAR
 3. MDR out, IR in
 - Ejecutar la instrucción
 4. IR_{OPE} out, MAR in, LEER
 5. R₁ out, Z in, ESPERAR
 6. MDR out, ADD, Y in
 7. Y out, R₁ in

Arquitectura de los Sistemas de Cómputo

10

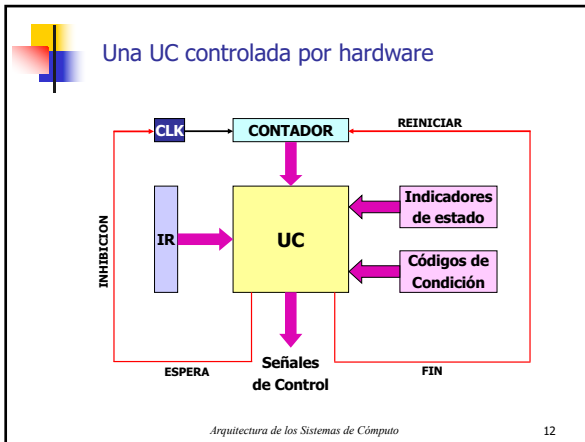


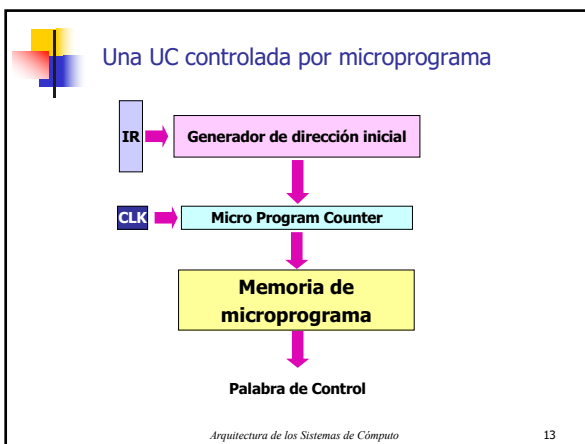
Secuenciamiento de microinstrucciones

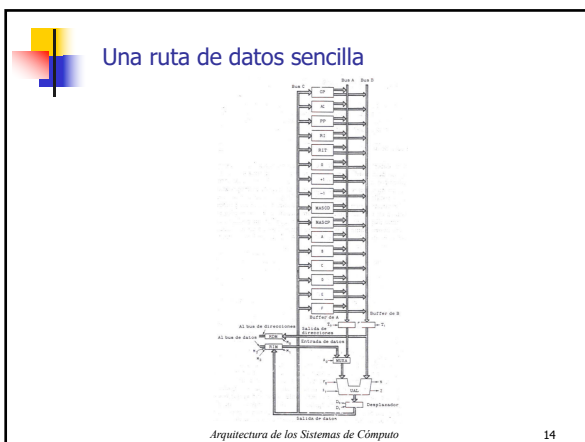
- El secuenciamiento de las microinstrucciones puede obtenerse por
 - Hardware (soluciones "cableadas")
 - Microprogramación (Wilkes, 1951)

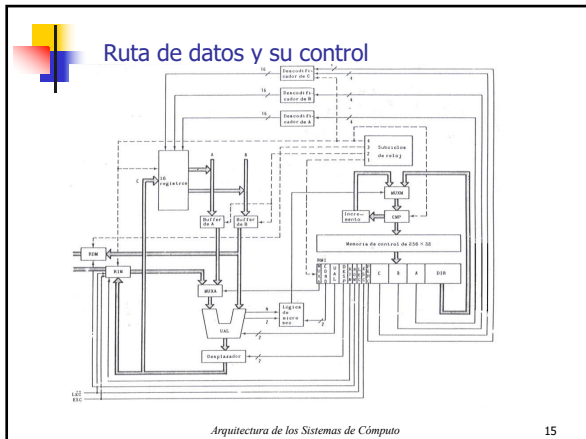
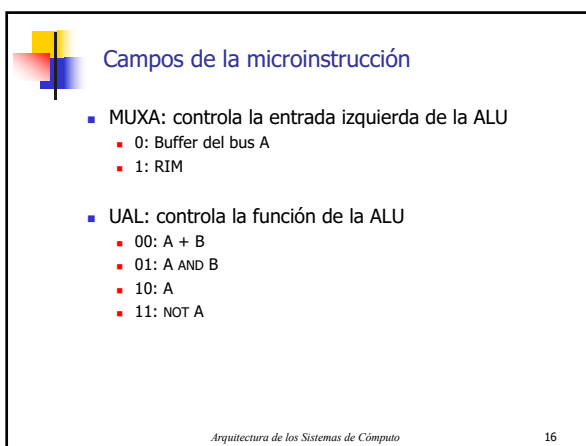
Arquitectura de los Sistemas de Cómputo

11







[illegible]





Campos de la microinstrucción

- RIM: Carga el RIM desde el desplazador (RDM)
- RDM: Carga el RDM desde el buffer de B
- LEC: Petición de lectura de memoria
- ESC: Petición de escritura de memoria
- PERC: Control de almacenamiento en MA
 - 0: No o deshabilitado
 - 1: Si o habilitado



Campos de la microinstrucción

- C: Selecciona el registro en el que se almacenará
- B: Selecciona la fuente del bus B
- A: Selecciona la fuente del bus A
 - En todos los casos un código de 0000 (CP) a 1111 (Reg. F)
- DIR: Dirección de la próxima microinstrucción



Cronología de las microinstrucciones

- 1: Carga de la microinstrucción en el registro de microinstrucción (RMI)
- 2: Salida de los contenidos de los registros a los buses A y B
- 3: Cálculo y eventual almacenamiento en el registro de direcciones de memoria (RDM o MAR)
- 4: Almacenamiento del contenido del bus C en la memoria de anotaciones y en el registro de datos de memoria (RIM o MDR) si es necesario



Una máquina convencional (ISA)

0000 xxxx xxxx xxxx	CARD	ac := m[x]
0001 xxxx xxxx xxxx	ALMD	m[x] := ac
0010 xxxx xxxx xxxx	SUMD	ac := ac + m[x]
0011 xxxx xxxx xxxx	RESD	ac := ac - m[x]
0100 xxxx xxxx xxxx	SPOS	if ac >= 0 then pc := x
0101 xxxx xxxx xxxx	SCERO	if ac = 0 then pc := x
0110 xxxx xxxx xxxx	SALTA	pc := x
0111 xxxx xxxx xxxx	CARC	ac := x
1000 xxxx xxxx xxxx	CARL	ac := m[pp + x]
1001 xxxx xxxx xxxx	ALML	m [pp + x] := ac
1010 xxxx xxxx xxxx	SUML	ac := ac + m[pp + x]
1011 xxxx xxxx xxxx	RESL	ac := ac - m[pp + x]

Arquitectura de los Sistemas de Cómputo

21



Una máquina convencional (ISA)

1100 xxxx xxxx xxxx	SNEG	if ac < 0 then pc := x
1101 xxxx xxxx xxxx	SNCERO	if ac <> 0 then pc := x
1110 xxxx xxxx xxxx	LLAMA	pp := pp - 1; m[pp] = pc; pc = x
1111 0000 0000 0000	APILAI	pp := pp - 1; m[pp] := m[ac]
1111 0010 0000 0000	DPILAI	m[ac] := m[pp]; pp := pp + 1
1111 0100 0000 0000	APILA	pp := pp - 1; m[pp] := ac
1111 0110 0000 0000	DPILA	ac := m[pp]; pp := pp + 1
1111 1000 0000 0000	RETOR	pc := m[pp]; pp := pp + 1
1111 1010 0000 0000	INTERC	tmp := ac; ac := pp; pp := tmp
1111 1100 yyyy yyyy	INCPP	pp := pp + y
1111 1110 yyyy yyyy	DECPP	pp := pp - y

Arquitectura de los Sistemas de Cómputo

22



Un programa


```
Program InnerProduct (output);
{Asigna valores iniciales a dos vectores x e y de 20 elementos
cada uno y luego calcula su producto interno o producto escalar}

const max = 20;

type SmallInt = 0..100;
vec = array[1..max] of SmallInt;
var k: integer;
    x, y: vec;
```

Arquitectura de los Sistemas de Cómputo

23



Un programa

```
function pmul(a, b: SmallInt): integer;
(Multiplica sus dos parámetros mediante
sumas sucesivas)

var p, j: integer;
begin
  if (a = 0) or (b = 0) then
    pmul := 0
  else
    begin
      p := 0
      for j := 1 to a do
        p := p + b;
      pmul := p;
    end
  end;
end;
```

{0: reserva espacio de pila para p y j}

{1: si cualquiera es 0 el resultado es 0}

{2: la función entrega 0}

{3: da valor inicial a p}


{4: suma de b a p a veces}

{5: realiza la suma}

{6: asigna el resultado a la función}

Arquitectura de los Sistemas de Cómputo

24




Un programa

```
procedure inner (var v: vec; var ans: integer);
{Calcula el producto interno}
var sum, i: integer;
begin
  sum := 0;
  for i := 1 to max do
    sum := sum + pmul (x[i], v[i]);
  ans := sum
end;

begin
  for k := 1 to max do
    begin
      x[k] := k;
      y[k] := pmul (2, k) + 1;
    end;
    inner (y, k);
    writeln (k)
  end.
```

Arquitectura de los Sistemas de Cómputo

25



Traducción parcial a ensamblador ISA

```
K = 4020
X = 4000
Y = 3980
A = 4
B = 3
P = 1
J = 0
V = 4
ANS = 3
SUM = 1
I = 0
JUMP      MAIN
```

Arquitectura de los Sistemas de Cómputo

26




Traducción parcial a ensamblador ISA

PMUL:	DECPP	2	/0		SCERO	L2	/A = 0; no se ejecuta
	CARL	A	/1	L1:	CARL	P	/5
	SNICERO	ANOTZ	/SALTA SI A <= 0		SUML	B	/AC := P + B
	CARC	0	/2		ALML	P	/P := P + B
	SALTA	DONE	/RETORNA 0		CARC	1	/Comprueba fin iteración
ANOTZ:	CARL	B	/AC := B		SUML	J	/AC := J + 1
	SNICERO	BNOTZ	/SALTA SI B <= 0		ALML	J	/J := J + 1
	CARC	0	/2		RESL	A	/AC := J - A
	SALTA	DONE	/RETORNA 0		SNEG	L1	/Bifurca si J < A
BNOTZ:	CARC	0	/3		SCERO	L1	/Bifurca si J = A
	ALML	P	/P := 0	L2:	CARL	P	/6
	CARC	1	/4		DONE:	INCP	2
	ALML	J	/J := 1		RETOR		/RETORNA
	CARL	A	/¿Se ejecuta la iteración?				
	SNEG	L2	/A < 0; no se ejecuta				

Arquitectura de los Sistemas de Cómputo

27



La pila

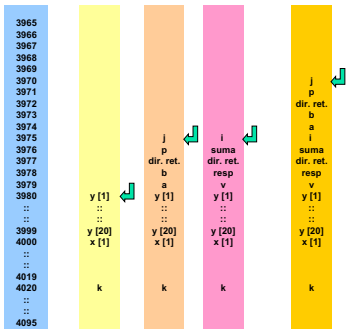



Diagram illustrating the stack structure and data flow during execution. The stack grows downwards from higher memory addresses (3965) to lower addresses (4095). The diagram shows four states: Estado Inicial, mulp, inner, and mulp desde inner. Arrows indicate the movement of data between these states, specifically showing the push and pop of registers like j, p, dir, ret, b, a, i, suma, dir, ret, resp, v, y, x, and k.

Arquitectura de los Sistemas de Cómputo

28



Manejo de la pila

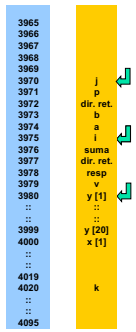


Diagram illustrating the stack structure and data flow during execution. The stack grows downwards from higher memory addresses (3965) to lower addresses (4095). The diagram shows the initial state and the state after the 'mulp' operation. Arrows indicate the movement of data between these states, specifically showing the push and pop of registers like j, p, dir, ret, b, a, i, suma, dir, ret, resp, v, y, x, and k.

Arquitectura de los Sistemas de Cómputo

29

