

OC: grupo 1/16a

decimal: 0 - 16 = todos son sist. posiciones

Computadora ejecuta programas Almacenados en memoria en binario → Es lo + fácil de hacer. N° enteros sin/sin signo - Reales con signo - Decimales identificar

3/7 → Tarea prioritaria codificadas en binario (BCD) - caracteres

4/7 → Evolox fp. Sin signo, módulo y signo, (0,1 complemento a la base restringida) -

5/8 → Evolox fp. (por promox) (0,1 complemento a la base) - exceso

4/8 → Recuperatorio fp.

5/8 → Recuperatorio fp. N° enteros sin signo de 0 a + Rango de 0 a $2^8 - 1$

Ejemplo n=8 bits 0 0000000

128 10000000

255 11111111

$$\text{Teorema Fundamental de la Numeración} \quad N = \sum_{i=0}^{n-1} (d_i)_{10} \times (\text{base})^i$$

$$x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} B^{-1} + \dots$$

Ejemplo: base 10

$$3574: 3000 + 500 + 70 + 4$$

$$3 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

3 unidades de mil + 5 centenas + 7 decenas + 4 u.

$$3.1416: 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 6 \times 10^{-4}$$

3 u. + 1 decimal + 4 centésimas + 1 milésima + 6 diezmilésimas

Números en punto fijo: se considera q todos los números a representar tienen exacto lo misma cant de dígitos y la canto esto ubicado en el mismo lugar

Ej. bolígrafo

Rango: de donde hasta donde puede ubicar 1/los dígitos q canto
Resolv: df q 2 no consecutivos.

$$\text{Ej: } 0.00 \text{ a } 9.99 \text{ Resolv: } 0.01 \quad 9.99 - 9.98 = 0.01 \quad \{$$

NOTA

Operaciones del aritméticas \rightarrow Realizada x lo ALU

- Suma en binario: hoy acuerda el posiciones Ej: 11110 (1 y me lleva 1)
- Resta: 0-0=0 1-0=1 1-1=0 0-1=el de la siguiente me pone 2.

Bits de control (Bandas) Bits q' el procesador establece acuerdo al tipo de operación realizada.

- Permite tomar decisiones como: hacer o no una transferencia de control, determinar sobre el nº. Salta condicionales, nos lleva a una parte del código si/ito.

Z: vale 1 si el signo de las operaciones son todos 0

C(carry): en la suma vale 1 si hay acarreo en el bit + significativo. en la resta vale 1 si hay "borrow" hacia el bit + significativo mirando < al sustraendo.

$$\begin{array}{r} \text{Ej: } \begin{array}{r} 1001 \\ + 1010 \\ \hline 10001 \end{array} \quad \begin{array}{r} 1001 \\ - 1100 \\ \hline 1101 \end{array} \\ \text{Flag acarreo} \qquad \qquad \text{Flag carry} \end{array}$$

Sistema hexadecimal \downarrow : Base 16

Dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

$$\begin{aligned} \text{Ej: } 2CA.8_{16} &= 2 \times 16^3 + C \times 16^2 + A \times 16^1 + 8 \times 16^{-1} \\ &= 512 + 192 + 16 + 0.5 \\ &= 714.5_{10} \end{aligned}$$

$$(2 \times 16 = 2^4 \times 16 = 4 \text{ bits}) \quad \begin{array}{c} 16 \\ 0 \quad 0000 \\ 1 \quad 0001 \\ 2 \quad 0010 \\ 3 \quad 0011 \\ 4 \quad 0100 \\ 5 \quad 0101 \\ 6 \quad 0110 \\ 7 \quad 0111 \\ 8 \quad 1000 \\ 9 \quad 1001 \\ A \quad 1010 \\ B \quad 1011 \\ C \quad 1100 \\ D \quad 1101 \\ E \quad 1110 \\ F \quad 1111 \end{array} \quad \begin{array}{c} 2 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{array} \quad \begin{array}{c} 16 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{array}$$

Si me faltó, higrelo en los grupos de 4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
2	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111		
3	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111			
4	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111				
5	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111					
6	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111						
7	0111	1000	1001	1010	1011	1100	1101	1110	1111							
8	1000	1001	1010	1011	1100	1101	1110	1111								
9	1001	1010	1011	1100	1101	1110	1111									
A	1010	1011	1100	1101	1110	1111										
B	1011	1100	1101	1110	1111											
C	1100	1101	1110	1111												
D	1101	1110	1111													
E	1110	1111														
F	1111															



$$\begin{array}{r} 434,75 \\ \hline 451 \\ 256 \\ 175 \end{array} \quad \begin{array}{r} 0,45 \\ \hline 451 \\ 256 \\ 175 \end{array}$$

BCD: F/S preferencias: los nº se codifican usando un byte por dígito. Se dice q' el nº está desempaquetado. Decimal: En rotulo, se reservan 4 bits x dígito. Se dice q' el nº está empaqueado codificado en binario.

sin signo, pl. emisor y receptor c/ c/c relaj.

Las 4º d. difieren al dígito.

Ejemplo: 834 = 11111000 11110111 11110100

F8 F3 F4

Con signo: +834 = 11111000 11110111 11110100
F8 F3 C4

C = Signo + 1101

D = Signo -

Los 4 bits q' ocupan el último dígito son reemplazados x el signo

-834 = 11111000 11110111 11110100
F8 F3 D4

Empaqueado q' signo:

+834 = 1000011 01001100 083 4C

-834 = 0000011 01001100 083 4D

$$\begin{array}{r} \text{Suma en BCD} \quad 41 \quad 0100 \quad 0001 \\ \hline + 22 \quad 0010 \quad 0010 \\ \hline 63 \quad 0110 \quad 0011 \end{array} \quad \left. \begin{array}{l} 0100 \\ 0010 \\ 0010 \end{array} \right\}$$

$$\begin{array}{r} 15 \quad 0001 \quad 0000 \\ 27 \quad 0010 \quad 0001 \\ \hline 42 \quad 0011 \quad 0001 \\ 6 \quad 0001 \quad 1100 \\ \hline \rightarrow 0110 \end{array} \quad \begin{array}{r} 0001 \quad 0000 \\ 0010 \quad 0001 \\ 0011 \quad 0001 \\ 0001 \quad 1100 \\ \hline 0110 \end{array}$$

Nivel de lógica digital

- Un circuito digital es en el q' están presentes dos valores lógicos.
- Componentes son dispositivos electrónicos q' pueden realizar funciones q' estos 2 valores lógicos.
- Componentes básicos: AND, OR, NOT, NAND, NOR, XOR

Componentes simbolo y descripción funcional.

		
NOT	NAND	NOR
$A \mid X$	$A \mid B \mid X$	$A \mid B \mid X$
0 1	0 0 1	0 0 1
1 0	0 1 0	0 1 0
	1 0 1	1 0 0
	1 1 0	1 1 0

	
AND	OR
$A \mid B \mid X$	$A \mid B \mid X$
0 0 0	0 0 0
0 1 0	0 1 1
1 0 0	1 0 1
1 1 1	1 1 1

XOR XNOR

- 2 entradas = salida 0
 " " 1
 " \Leftrightarrow 1 " \Leftrightarrow 0



Algebra de Boole: leyes matemáticas q' gobiernan el comportamiento de las operaciones el. bits y describen circuitos

De Morgan: $\bar{A} \cdot \bar{B} = \bar{A} + \bar{B}$ / $\bar{A} + \bar{B} = \bar{A} \cdot \bar{B}$

$$\bar{\bar{A}} = A$$

Identidad: $1 \cdot A = A$ / $0 + A = A$

Nulla: $0 \cdot A = 0$ / $0 + A = A$

Idempotencia: $A \cdot A = A$ / $A + A = A$

Inversa: $A \cdot \bar{A} = 0$ / $A + \bar{A} = 1$

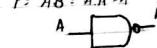
(Comutativa): $A \cdot B = B \cdot A$ / $A + B = B + A$

Asociativa: $(AB) \cdot C = A(BC)$ / $(A+B)+C = A+(B+C)$

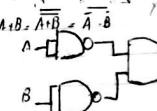
Distributiva: $A+B \cdot C = (A \cdot B)+(A \cdot C)$ / $A \cdot (B+C) = AB+AC$

Absorción: $A \cdot (A+B) = A$ / $A + A \cdot B = A$.

Ejemplo: • construir un NOT q' NAND $F = \bar{A} \cdot \bar{\bar{A}} = \bar{A}$



• Construir OR con NAND $F = A+B = \bar{\bar{A}} \cdot \bar{\bar{B}} = \bar{A} + \bar{B}$



• Escribir la tabla de verdad p/ la fr

• Dibujar una AND p/ cada término q' tiene 1 en la columna de salida.

• Invertir E necesario

• Unir todas las AND a una OR

Recoratorios

- En una AND, basta q: una de sus variables de E sea 0 p/q: lo fijo.
- En una OR,
-

Punto ①

1- Representar en el sistema BCS o B bits

1- 1000 0001	35.000 0011	No pude representar -253, 256, -1, +8, +
2- 1000 0001	35.000 0011	-127, +128, -199, -256, -100, 0.5, 1.25
3- 1000 1000	x9 excede los 8 bits o es negativo	0.00000000

137 = 1000 1001

2- Interpretar cadenas de 8 bits en BCS

0000 0000	1000 0000	1100 1111	1111 1111	0111 1111	1000 1111
0000 0000	1000 0000	1100 1111	1111 1111	0111 1111	1000 1111

3- 11111111 1111 00000000 Range: 0.00000000

4- Interpretar con el punto 3

10.000.00000	10.000.00000	10.000.00000	X000.00000	10.000.00000	10.000.00000
11.111111	11.111111	11.111111	0000.00000	11.111111	11.111111

5- 0

0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000

BCS, Col. Col2, n=1

- Técnica de complemento: Complemento de N dada N-A
- Complemento = V de (N-A): $N - (N-A) = A$

Close ② → Número de signo →

- Representar en BCS. Con n bits, 1 bit representa el signo y n-1 bits.
- Modulo y signo.

Ej: 10010 = +2
1010 = -2

- El bit extremo izquierdo representa solo el signo.
- Los bits 0 a n-2 lo mapean
- 0: positivo 1: negativo
- Range $-(2^{n-1}-1) \rightarrow +(2^{n-1}-1)$
- Doble representación del 0. → 1000 y 0000

- Complemento a 1 de m: $2^{n-1} - m$

- Complemento a 2 de m: $2^n - m$

Representar en Col2: 0+1-

- Ej: +32: 0010 0000
- 32: 1101 1111
- +7: 0000 0111
- 7: 1111 1000

$$\begin{aligned} n &= 8 \text{ bits} & N^0 &= \left\{ \begin{array}{l} 11111111 \\ 10000000 \end{array} \right. \\ && N^1 &= \left\{ \begin{array}{l} 11111111 \\ 10000000 \\ -(2^{n-1}-1) \end{array} \right. \\ && N^2 &= \left\{ \begin{array}{l} 01111111 \\ 00000000 \\ +(2^{n-1}-1) \end{array} \right. \\ && N^3 &= \left\{ \begin{array}{l} 00000000 \end{array} \right. \end{aligned}$$

• Dada una cadena de bits, ¿que valor representaría si lo interpreto en Col2?

$$+0110 0000 = 1 \times 2^6 + 1 \times 2^5 = 64 + 32 = 96$$

→ Ambas formas

$$+ Col2 del n^0 y obtengo el + = 1110 0000 = -32$$

NOTA

M y E representados en BCS, BCS, C02, C02, Exeo
mantissa matica

Símbolos del sg formato o pto flotante

Mantisa	B0S	Exponente	B0S
14 bits		4 bits	
entre		entre	
82.5 f de fp	Mantisa: 1111 × 2 ¹¹¹¹	Rango: 15 × 2 ¹⁵	→ punto representable
de exponente			
Mantisa 0.		Resolviendo extre. sup [0, ..., 15 × 2 ¹⁵] [0, ..., 49152]	
" (0000)000000000000			
" inf. R: (1-0) × 2 ⁻¹			
"			
R: (15-1) × 2 ¹⁵ : 1 × 2 ¹⁵			
"			

Mantisa y exponente en C02

Mantisa	C02	Exponente	C02
14 bits		6 bits	
flotante		flotante	
flotante		flotante	
82.5 f de fp	Mantisa: 0111 × 2 ⁰¹¹¹ = +7 × 2 ¹	Rango: -8 × 2 ¹ ... +7 × 2 ¹	
de exponente			
Mantisa 0.	Mantisa: 0100 × 2 ⁰¹¹¹ = -8 × 2 ¹		
"	Rango: [-8 × 2 ¹ ... +7 × 2 ¹]		
"	Resolviendo el extremo superior: R: 7 × 2 ¹ = 1 × 2 ²		
"	" en el origen: R: (1 × 2 ⁰) = 1 × 2 ⁰		

Mantisa formanaria todos los bits sin formaciones

Mantisa	B0S	Exponente	C02
23 bits		8 bits	
fraccionaria			7
1 bit signo			

* Resolvemos *

Máximo positivo 0,111...111 × 2¹¹¹¹¹¹¹¹ = +(12⁻²³) × 2¹¹¹

Mínimo + (f0) 0,000...001 × 2¹⁰⁰⁰⁰⁰⁰⁰ = -(2⁻²³) × 2⁻¹²⁸

Máximo negativo (f0) + 0,000...001 × 2¹⁰⁰⁰⁰⁰⁰⁰ = -(2⁻²³) × 2⁻¹²⁸

Mínimo negativo 1,011...111 × 2¹⁰¹¹¹¹¹¹ = -(2⁻²³) × 2⁻¹²⁸

Formato final signo 01 exponente 8 Mantisa 31

Mínimo negativo 1,0111111,11111111111111111111

Normalización: 40.10⁻⁴ = 4 × 10⁻⁴ = 0.4 × 10⁻³ = 400 × 10⁻⁷

Existen → valores de mantisa y exponente pl/representa un mano n° ±0,1111...111 × 2¹¹¹ / el sbg. de tener un rango por de valores de mantisa y exponente pl/ unív. se indica la formación. → solo pl/ formaciones

Si el obj. calcular, la mantisa formanaria se define una

largo de n bits en la parte entera

solución → 2ⁿ⁻¹ - 1

solo el dígito

Exemplo

Mantisa	Exponente	Rango
23 bits	7 bits	
fraccionaria		
1bit signo		
Mantisa redonda		

Max + 0,111...111 × 2¹¹¹¹¹¹¹¹ = +(12⁻²³) × 2¹¹¹

Mín. + 0,000...001 × 2¹⁰⁰⁰⁰⁰⁰⁰ = +(12⁻²³) × 2⁻¹²⁸

Mín. negativo + 0,000...001 × 2¹⁰⁰⁰⁰⁰⁰⁰ = -(12⁻²³) × 2⁻¹²⁸

Min - 1,011...111 × 2¹⁰¹¹¹¹¹¹ = -(12⁻²³) × 2⁻¹²⁸

Rango F: -(2⁻²³) × 2¹¹¹, -(0,5) × 2⁻¹²⁸] [- (0,5) × 2⁻¹²⁸, (2⁻²³) × 2¹¹¹]

Cases especiales

1. Exponente: 255 o 2047, M ≠ 0 \Rightarrow No N "Not a number"
2. E.p.: 255/2047, M = 0 \Rightarrow zero
3. E.p.: 0 M ≠ 0 \Rightarrow zero
4. E.p.: 0 M = 0 \Rightarrow Desnormalizado: n = pequeño dígito que no ingresa
 $\pm 0, \text{mantisa} = s \cdot p^{2^{-16}} / \text{mantisa}_0$

Operaciones aritméticas en pto flotante

- 1º solo igualar los exponentes \rightarrow Hacer coincidir el p/exponente en ambos.
- Suma y resta Cuando los exponentes son \neq , se resta el menor y se da el resultado. M es el de $\pm N \cdot 2^{\text{exponente menor}}$.

Multiplicar y dividir: Suma y resta exponentes:

- Multiplicar y dividir mantisas y expon.

- Normalizar

- Redondear

Todos los dígitos intermedios se borran.
Lo que queda se almacena.

$$\begin{array}{l} x = x_s \cdot B^{x_e} \\ y = y_s \cdot B^{y_e} \\ \hline x \cdot y = (x_s \cdot y_s) \cdot B^{x_e + y_e} \end{array}$$

$$x \cdot y = (x_s \cdot y_s) \cdot B^{x_s + y_s}$$

$$\frac{x}{y} = \left(\frac{x_s}{y_s} \right) \cdot B^{x_s - y_s}$$



• Conjunto de portas para conectar una sola salida f a la salida de las portas de ese instante. La salida de la f tiene que ser una combinación de las salidas de las portas.

Clase 4: circuitos lógicos combinacionales y secuenciales:

- Repetir o voltear logicas en E.
- Si cambia la E, cambia S \rightarrow f se cambia.
- Los val. ordenados de las E, cambian la f, pero no tienen \Rightarrow no influyen los v. de S.
- Se representa mediante tabla de verdad o diagrama de真理表 (verdadero, falso).

T, o, o

• Circuito multiplicador de 8E/8E: tiene muchas f y una sola S.

P/selección E, tiene 3 líneas de selección o selector de datos.

Ej.: televise, selecciona, el vales E p/f

ALU: hace operaciones

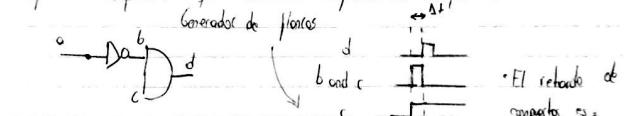
S se hace + y -, pedo hacer el resto de op.

Internet: compartimos el ancho de banda.

Simultáneamente el mismo IP y se distribuye el los dispositivos.

• Existe tab multiplexor en tiempo.

• Respuesta temporal: lo q. tardan en responder los componentes.



• El rebote de componente es:

Hace en da el rebote de componente.

P/cada una señal q' sea un pulso de para diox q' se sincronizan o sin un reloj p/ otros circuitos.

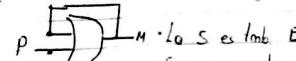
NOTA

• PC: almacena / procesa
• Se addressan circuitos separados de registradores
estados anteriores.

Así surgen los circuitos secuenciales:

- Dependen de su E anterior y E interno del circuito
- S depende de E

¿Cómo se almacena en reloj lógico?



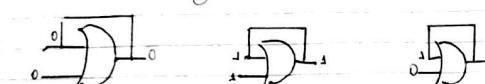
• La S es lmb. E

• En nega cuando cambia de uno S

transporta info. o una E

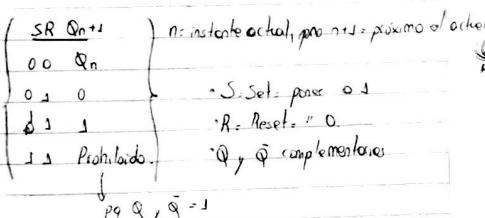
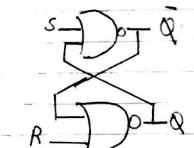
• La ecua lógica $M = M + P$.

Ejemplo:



$M = M + P = 0:0:0 \quad M = M + P = 1:1:1 \quad M = M + P = 1:0:1$

Tipo: son bistables.
FLIP-FLOP SR: os. secuenciales. 2 S complementarios



Los secuenciales se pueden clasificar en:

Asternarios: el cambio de la S, cruce al cambio de la E + S anterior.

Sinestriarios: la S no cambia en el momento, sino cambian a

partir de una señal de reloj.



Reloj: "señal especial"
• Temporiza eventos del sist.

T largo: frecuencia baja } ciclo de tijo: q' /
T corto: frecuencia alta } del periodo lo corto
ciclo de tijo esto alto o bajo.

FLIP FLOP D:
 $\downarrow S \quad R = \bar{D}$

FLIP FLOP JK:

FLIP FLOP T: Q combio de 0 a 1
 $\downarrow S, \bar{J}, \bar{K}$
• 1 o 0 en cada pulso
de la E de T

Recordando un bit: pl. recordar un bit en el registro
hago q' usar un Flip flop D

Selección y operaciones:
• Identifica
una celda.

Contador módulo 8: 3 FLIP FLOP: rotador módulo 8.

• Operación de los 3 ciclos de los 3 flip-flops para dar resultado de 0 a 7
• A los 3 flip-flops se les da la señal de reloj.

Circuito combinacional
Entrada \rightarrow Circuito combinacional \rightarrow salida
Entrada \rightarrow Circuito combinacional \rightarrow Salida
AND Memoria

	S	R	Q	Q'
Fixo	1	0	1	0
Memoria	1	1	Memorizar, vale el E y sale Limpio	
Limpio	0	1	0	1
Memoria	1	1	1	0
	0	0	Indefinible	

	OR			
	S	R	Q	Q'
Set	0	1	1	0
Memoria	0	0	Vale anterior	
Resetar J	0			
Memoria 0	0			
Fixo 1	1			
			Indefinible	

Flip flop con reloj \rightarrow Pulso del reloj:
Pulso bajo=0, depende del reloj.

	OR			
	Q	S	R	Q'
	0	0	0	0
	0	0	1	0
	0	1	0	1
	0	1	1	Indefinible
	1	0	0	1
	1	0	1	0
	1	1	0	
	1	1	1	0

FLIP FLOP D \rightarrow Reloj de entrada

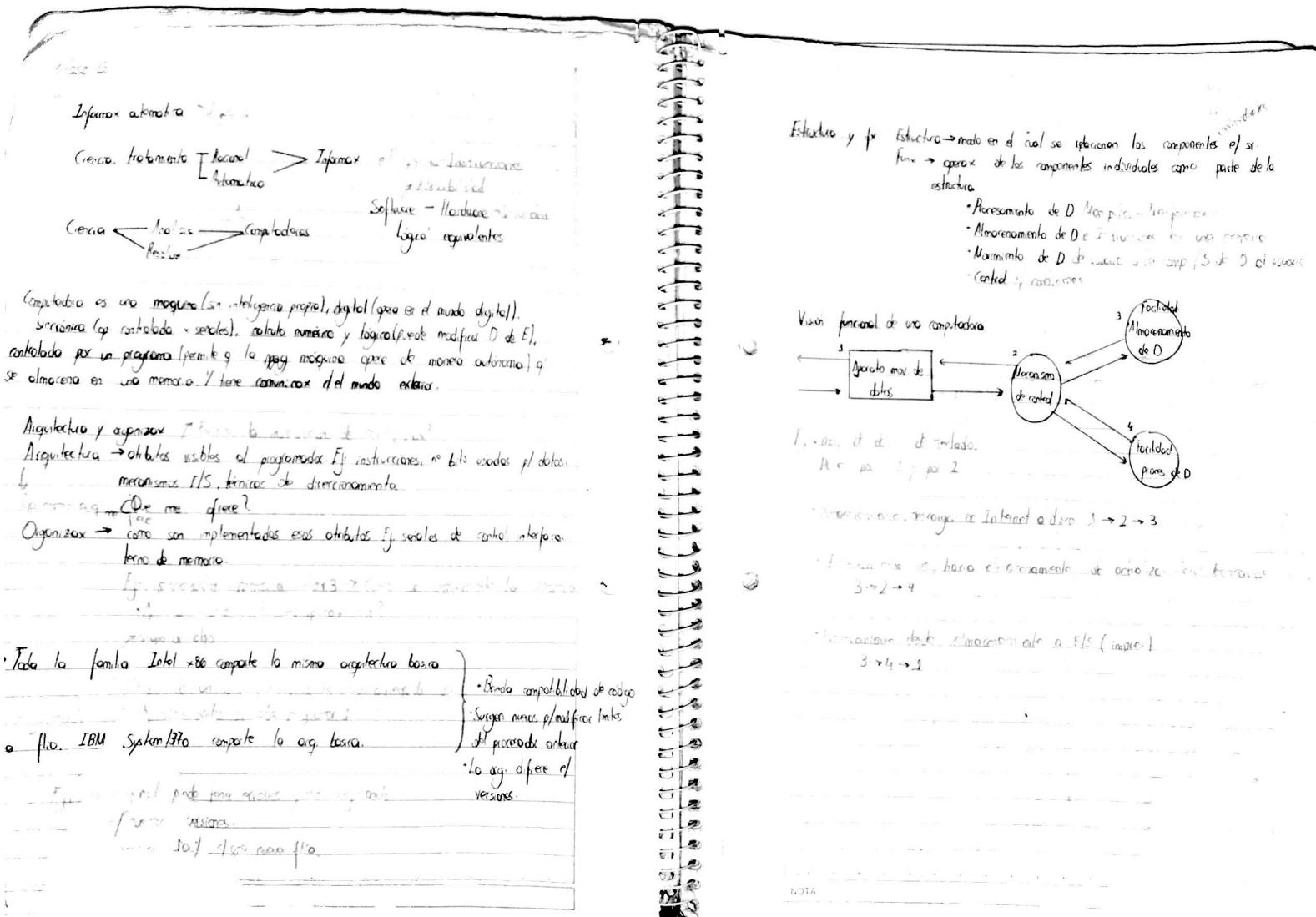
D	Q	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

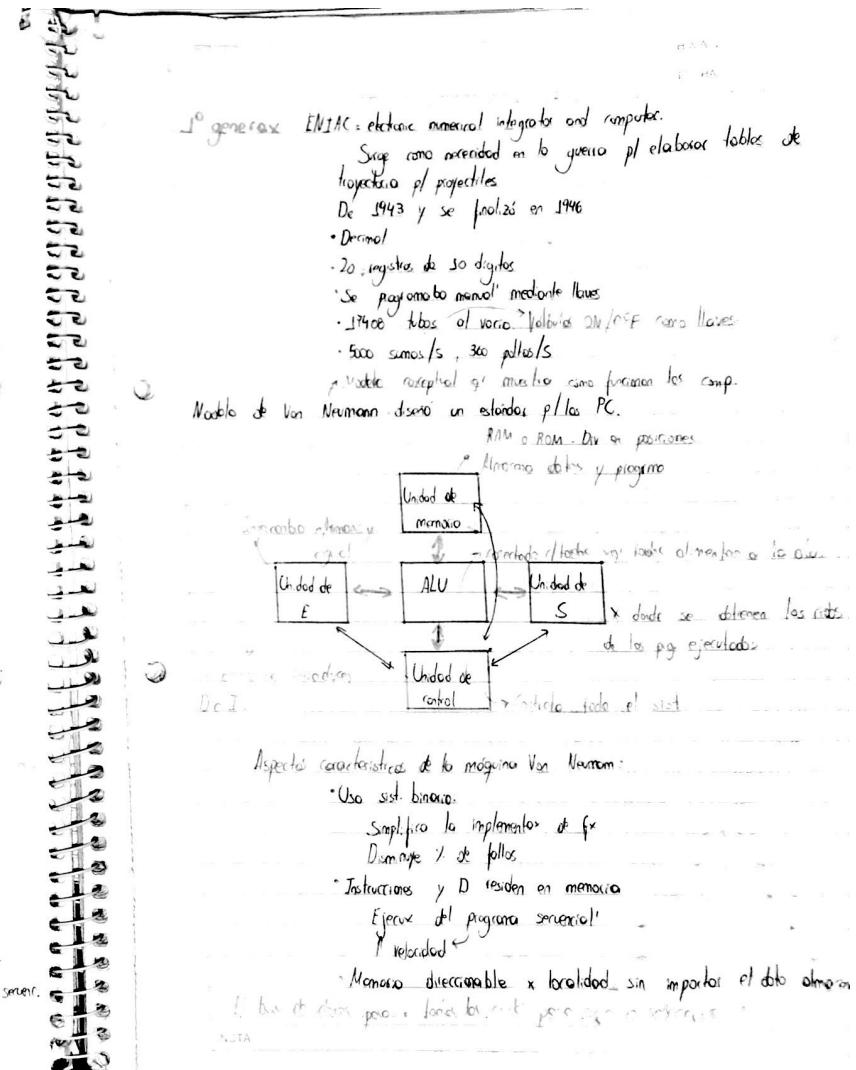
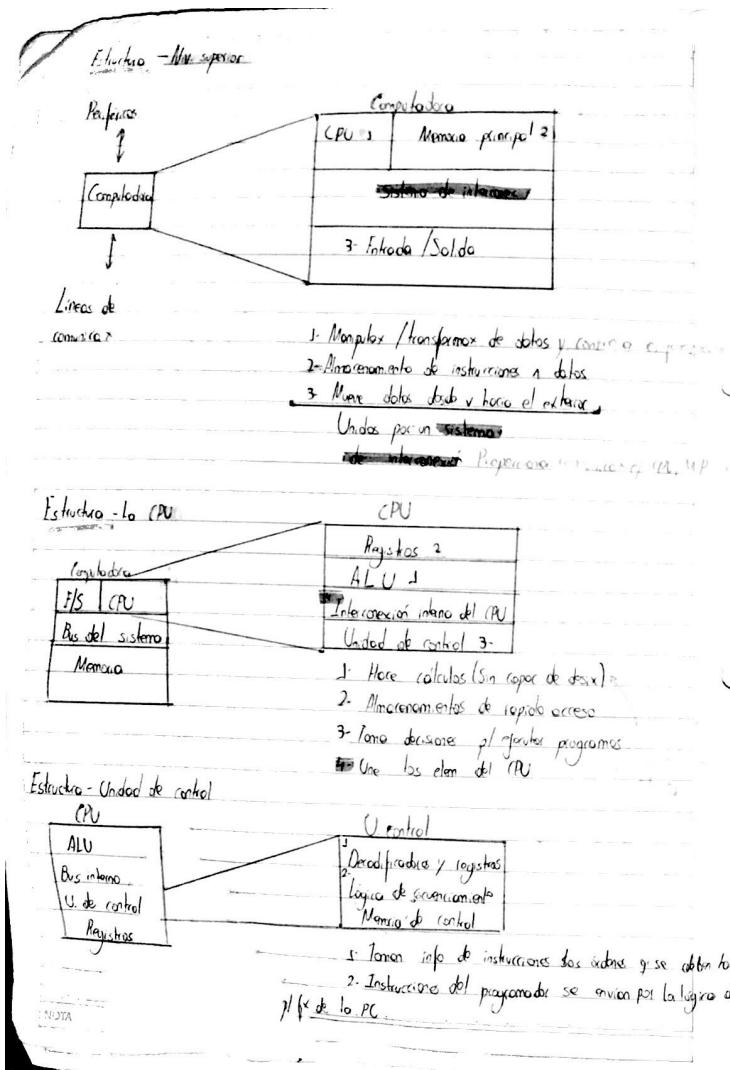
FLIP FLOP JK \rightarrow Reloj

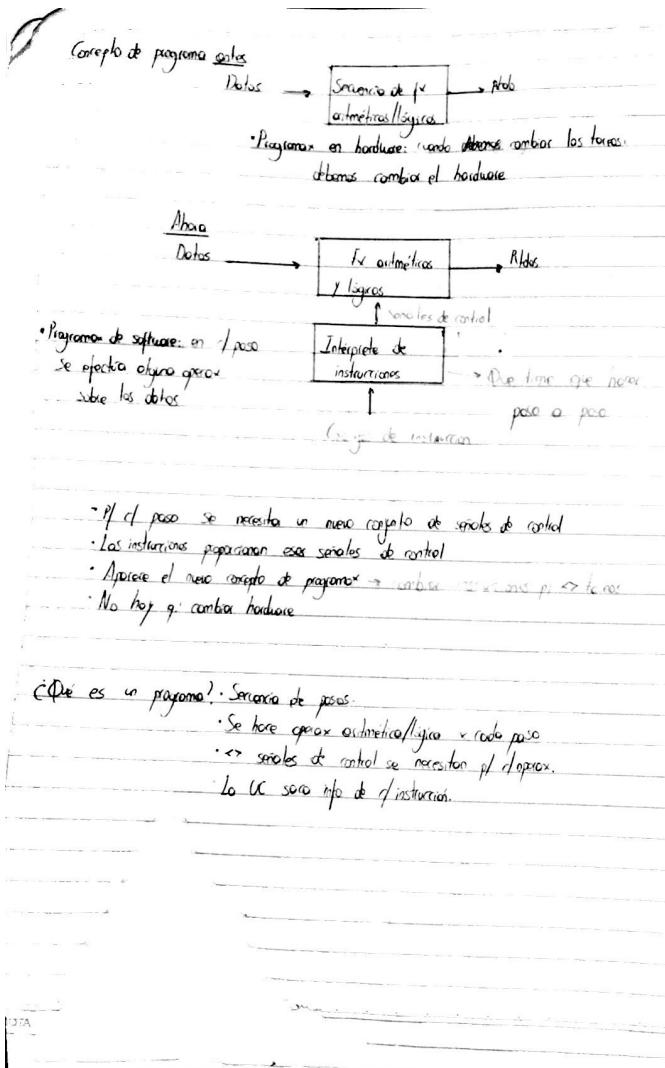
J	K	Q	Q(t+1)
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

FLIP FLOP T \rightarrow Toggle lo contrario

T	Q	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0







IAS: Institute of Advanced Study • Memoria / 4096 palabras de 40 bits

- Nº C2
- 2 inst de 20 bits
- Set de registros
 - Buffer de memoria (MBR)
 - direcciones de memoria (MAR) Direccion de memoria
 - Controles 8 bits de código de operación
 - Instrucción y buffer de instrucción
- Dirección de la memoria para ejecutar instrucciones + controlador de programa (program counter) PC
- Mecanismo ejecutando u otros temporizadores de lectura y escritura
- Sumador y multiplicador AC
- Divididor y multiplicador MC

UNIVAC I: Universal Automatic Computer

- 1º PC comercial
- Puede usar un compilador pt. traducir idioma de programación a máquina
- Directorio de 12 dígitos x palabra
- Principal almacenamiento: set de discos magnéticos q se giran hacia otras o adelante
- procedimiento de comprobación de errores
- Memoria de tareas de trabajo de memoria y tareas o volúmenes de trabajo

LBM

- Tipos de procesamiento: off-line y periodo:
- 1953: 701 → 1º PC q guardamos almacenados de IBM
- Aplicaciones científicas
- 1955: 702 → aplicaciones de gestión
- 1º serie de computadoras: 700/1000

NOTA

2º generación: transistores

- Cambian los tubos al vero

Transistor

- más pequeñas y bocas

menos calor

- Dispositivos de 5 soldadura

Intensidad mayor que de 1 mA.

3º generación: circuitos integrados

Circuitos integrados

- Mayor o menor escala

+ de los componentes en un chip

- a media escala

100-3000 componentes

- gran escala

3000-10000 componentes

- muy gran escala

10000-100000 millones de componentes

Serie IBM 600

- Instrucciones similares

E/S similares

- > velocidad

No 1 de E/S paralelo

- > memoria

> precio

DEC PDP-8

1º minicomputador

No necesitaba oscilador externo

App monostable y ROM

Estructura de bus + bocinas

Algo por los computadores; se fue desarrollando la fabricación de memorias

Memoria Semiconductora

1970

1º memoria 1/256 bits

Capacidad duplicada x año.

Lectura no destructiva

Microprocesadores: Intel

1971 4004 - 1º microprocesador de 4 bits

Todo lo del CPU en un solo chip

1974 8080 - 1º microprocesador de uso general

Intercanvier de en sist de computo *paralelo*

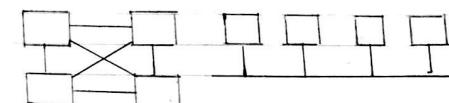
constituido por 3 subsistemas

• CPU
• Memoria

Los componentes deben comunicarse el/ si

E/S

El concepto de buses tiene canales independientes y de rápido comunicar



- Canales independientes el/ <> dispositivos
- Comunicación mediante un medio compartido.

Bus: • Camino de comunicar q' conecta 1 o + dispositivos local: "broadcast". Negativo posicionamiento.

Bus serie: 1 bit por vez + arriba el bus = + info q' viaja

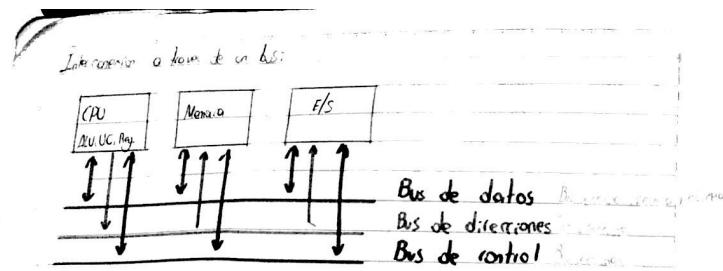
Bus paralelo: conjunto en serie

Transportar datos

No hay diferencia q' dato o instrucción

El ancho es un valor determinado de las prestaciones

• 8, 16, 32, 64 bits



- Direcciones:
- Si el bus es compartido por <> elementos, estos deben tener identidad <> direcciones
 - La dirección de memoria identifica una celda de memoria en lo q almacene info.
 - La lectura y escritura se plantean respecto de la PU

Bus de direcciones: Identifica el origen/destino de los datos

- La CPU necesita tener una instrucción de dato abierto
- El ancho de bus determina la máxima capacidad de memoria del sistema

Ej: 8-bit tiene bus de 8 bits dando un espacio de dirección de $2^8 = 256$ celulas de memoria.

Ej: 32-bit tiene bus de 32 bits dando un espacio de dirección de $2^{32} = 4.295 \cdot 10^9$

El tamaño de datos celula viene dado x el bus de datos

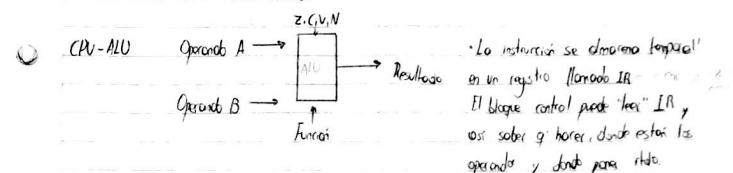
Bus de control: Info de control y temporizado

- Sentidos de escritura/lectura de memoria o E/S
- Sentidos de selección o habilitar
- Sentidos de reloj (clock)
- Sentidos de parada o interrupc.

- Componentes del hardware dedicado a I/F
- E/S: - Teclado
 - Monitor
 - Mouse
 - Impresora
 - Joystick

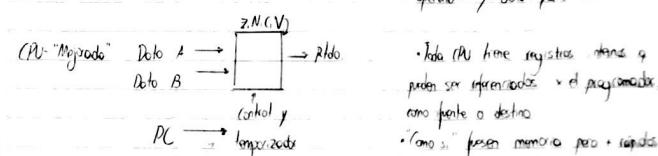
Almacenamiento - Memoria

- Disco (reg. de discos)
- Cintas (CD, DVD)



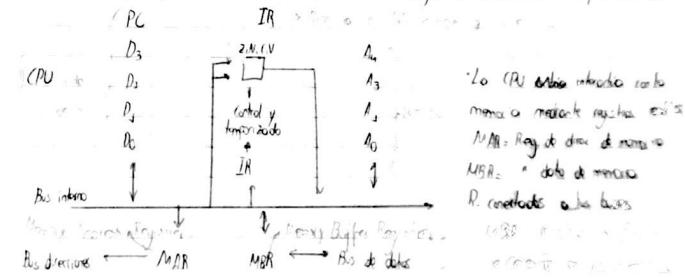
• La instrucción se almacenó temporalmente en un registro llamado IR - \rightarrow IR

• El bloque control puede leer IR y los sobre q hacer, donde están los operando y dest para rta.



• Todo CPU tiene registros internos q pueden ser referenciados x el programador como fuente o destino

• Como s. fueron memoria pero + rápida luego el almacenamiento temporal

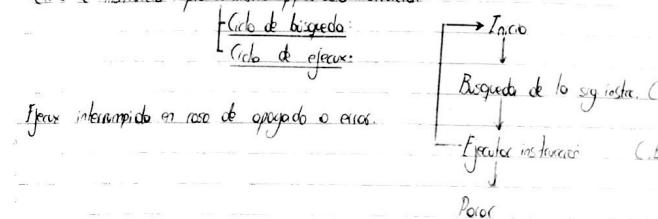


Todos trabajan x p/ obtener más direcc. p/ > flexibilidad

Clase 6. Ciclo de instrucción → surge de la ejecución de programas

- El procesamiento de instrucciones se divide en 2 etapas
 - Buscado: lee dato memoria
 - Ejecución: depende de la instrucción
- El procesamiento de instrucciones se divide en 2 etapas
 - Buscado: lee dato memoria
 - Ejecución: depende de la instrucción

Ciclo de instrucción: procesamiento p/ 1 sola instrucción



CICLO DE BÚSQUEDA y EJECUCIÓN

1. Al principio de todo ciclo, la CPU busca una instrucción en memoria
2. En la CPU hay un registro, llamado PC, q' tiene la dirección de la próxima instrucción a buscar
3. La CPU: dep. de buscar la instrucción, incrementa el valor del PC p/ buscar la sig. instrucción en memoria
4. La instrucción buscada se carga dentro de un registro de la CPU, llamado registro de instrucción (RI)
5. La instrucción está en la forma de un código binario q' especifica las órdenes q' forman la CPU
6. La CPU interpreta cada instrucción y lleva a cabo los órdenes requeridos

DIFERENTES CLASES DE INSTRUCCIONES

- CPU-MEMORIA: Datos q' pueden transferirse el memoria y CPU
 - Lectura
- CPU-E/S: datos q' pueden transferirse el CPU y E/S
- Procesamiento de datos: CPU efectúa operaciones aritméticas/lógicas en datos

Control: alterar la secuencia de ejecución de instrucciones

• Se da en forma de código

- EJEMPLO
- Cargar en el registro D el contenido de la posición de memoria 940.10
 - Sumar el contenido de la posición de memoria 941.10 al registro D y guardar el resultado en D
 - Almacenar el valor del registro D en la posición 942.10

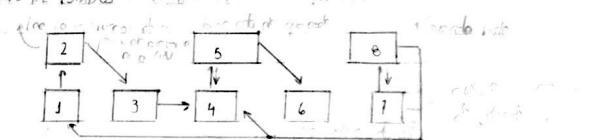
Consideremos q' el posix de memoria almacena 4 bits. Los 1º 4 bits indican lo q' se hace o realizar, los sig. 12 bits es la dirección.

• 0001.1 → cargar D desde la memoria

• 0110.2 → almacenar D en memoria

• 0101.5 → sumar D / un dato de memoria

DIAGRAMA DE ESTADOS → "Ciclo de ejecución"



Transiciones p/ cada dato

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

• 1 → 2: dato de memoria

• 2 → 3: dato de memoria

• 3 → 4: dato de memoria

• 4 → 5: dato de memoria

• 5 → 6: dato de memoria

• 6 → 7: dato de memoria

• 7 → 8: dato de memoria

• 8 → 1: dato de memoria

- Los E en la parte superior implican un intercambio c/ CPU y la memoria
- Los E de la parte inferior implican solo qd. internos en la CPU

Lenguaje Assembly

No resulta confortante pensar y escribir programas como una secuencia de nq. sino que es natural verlos como una secuencia de instrucciones. P/ ello se define un lenguaje q es un anglo de esos códigos comprensibles x un humano. A esos lenguajes se lo denomina assembly o lenguaje de ensamblar. El programa q entienda esos códigos se llama assembler.

Cierre 7 → formato de instrucción y modo de direccionamiento.

Elementos de una instrucción de máquina

- opcod /-Código de operación (Especifica lo q opera o realiza; es un código binario)
- Referencia del operando fuente, establece donde se encuentra el operando.
- Lo operando puede incluirse j o + operando fuente/o de t/.
- Referencia del destino para establecer dónde almacenar el resultado.
- Referencia de la sig. instrucción: le dice al CPU dónde buscar la sig. instrucción desp. de la ejecución de la instrucción anterior. En la mayoría de los casos se ubica a continuación de la instrucción actual.

• Los operandos fuente y resto pueden estar en memoria, registro CPU o disp. de I/O.

• Representación de instrucciones:

- Dentro de la computadora cada instrucción está representada mediante una secuencia de bits, que se divide en campos en correspondencia a los elementos q la componen.
- Este esquema se conoce como formato de la instrucción.
- Es difícil p/ el programador tratar el representación binaria, por lo tanto se usa una representación simbólica.
- Los códigos de operación se representan x medio de abreviaturas llamadas mnemónicas q indican lo q opera.

FORMATO DE INSTRUCCIÓN ADD 12 MA

• Los operandos también pueden representarse simbólicamente

Ej: mem, reg, mem/... → variable q opera. Reg: Operando. Mem: dirección. Instucción q opera el valor contenido en la posición de memoria llamado memoria, o en un registro denominado reg!

modo de operar → Referencia al operando → Ref. al operando

Ej: $x = x + y \rightarrow$ otro nivel lenguaje → Una instrucción q no regresa variables instrucciones de máquina. Expresa operaciones en forma concisa usando variables

• El lenguaje de máquina expresa las q. básicas usando mem. de datos y registros.

• Un lenguaje de alto nivel debe convertirse a un lenguaje de máquina. Un conjunto de instrucciones de máquina debe ser capaz de expresar cualquier instrucción de T. nivel.

• Clasificación de las instrucciones

- Almacenamiento de datos → operaciones aritméticas y lógicas
- Almacenamiento de D → transferencia dentro del sistema
- Instrucciones E/S → transf. de D al PC y mecanismos externos
- Control

Número de direcciones fija s/ la máquina

{ 2 direcciones p/ los operandos, una p/ almacenar el resultado y al la próxima instrucción
Add, Direccion, Disp1, Disp2, Dir. Fijo, Ins}

Máximo de 4 direcciones

• Muchos campos > espacio y + largo

NOTA

Moguna de 3 direcciones Add Dr.Res. D1Op1 D1Op2

- Direc más usada almacenada en CPU en el PC
- Tmb larga

Moguna 2 direcciones Add D1Op1, D1Op2 → Mem. tempor

- La q1 se move o muy lento
- dirección donde guarda res.

Moguna 1 dirección Add D1Op1

- Reg expandido en PV (acumulador)
- Instrucción pl. carga y descarga del acumulador
- Un grande y tmb en fija profund
- Inst. corta

Ejemplo $o = (b+c)^d - e$

3 direcciones	2 direcciones
add o, b, c	mv o, b
mul o, d	add o, c
sub o, e	mul o, d
sub o, e	sub o, e

1 dirección

b	c	d	e	o
add b				
add c				
mul d				
sub e				
sub e				

↓

Diseño del conjunto de instrucciones

- Modo q tiene el PC programador pl. controlar CPU
- Tener en cuenta - tipos de operaciones (nº y cantidad)
- tipo de datos (valores)
- formato de instrucciones: longitud, nº dirección, tempor. largo
- Registros: cant q se pueda referenciar
- direccionamiento: forma de especificar la dirección de un operando/instrucción

Tipos de operaciones

- Transferencia mem/buf/scr
- Add, sub, inc, dec, mul
- And, or, xor, not
- Guardar
- F/S, in, out
- Transferencia de control salto/bifurcas
- Control del sistema usadas x 50

Tipos de datos

- Direcciones
- Nº. fl. p. fijo/floating
- Controles IXII, BCN
- Datos ligeros

Modo de direccionamiento:

- Inmediato: se usa gran cant de bits pl. especificar origen. / simplifica esc.
- Se usa el modo de direccionamiento.
- Directo: si un operando no usa varia direc, para referirse a un registro. (Atrés + registro/bits)
- X registro
- Indirecto x memoria
- Indirecto x registro
- X desplazamiento
- Del start
- y procesada.

2º: especifica 1 o + operandos en forma implícita. Ej: reg2 = registros, cuando los desplazan bits en la instrucción, para no necesitar dirección hasta ejecutar programa, porque suficiente de D:

NOTA

MDD Inmediato Dirección Operando El operando se obtiene directamente de la memoria o mismo tiempo q la instrucción.

- No requiere una referencia extra a memoria de datos.
- Se usa para definir constantes y personalizar variables.
- Tamaño de operando limitado x el tamaño del campo de direccionamiento
- Aplic. de r^o más grande (no q se extienda q el abanico)

MDD Directo Dirección Operando Proporciona dirección de memoria

Memoria • Espacio limitado de direc. x razón de tamaños del campo.

• Uso: variable global; dirección constante al compilador.

MDD por registro Dirección Operando • = el directorio

Registers (RN) • Uso: - bits.

• Pasa registros y su resultado.

MDD por memoria Dirección Dirección 2 • 1/ una dirección - bit en la instrucción.

• se apunta a una dirección + bits

Memoria • = espacio de almacenamiento

• Multiples accesos a memoria

MDD indirecto por registro

<u>Registro</u>	<u>Dirección</u>	<u>Operando</u>
Registers (RN)	Memoria	• bits p/ un registro q pase de memoria

MDD x desplazamiento R, A Dirección Operando

R	A	Dirección	Operando
Registers	Registers	Memoria	Memoria

• Cambio directo e indirecto

• 3 modos de direccionamiento.

* Nota: Registro referenciado: PC, la dirección q lo indique se suma el campo de direc. p/ producir dirección efectiva. El campo de dirección se hace cargo de R2.

* Registro base: el registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.

Inmediato: se direcciona la memoria q un registro + un desplazamiento.

- = q el autor para se intercambien partes del registro y desplazamiento.
- Proporciona un mecanismo eficiente p/ realizar q's iterativas.
- Se usa un registro llamado índice (se incrementa o decrementa s/ index).

MDD del stack → Almacena local de tránsito de memoria. Lista abierta el último en entrar es el último en salir. Zona de memoria reservada.

Asociado q la pila o stack, hoy un registro apuntador.

• Pueden gestionar las llamadas y retornos de procedimientos, y pueden contenerse entre una forma alternativa de direccional memoria. C/L dirección conjunta de posiciones last-in-first-out.

No de élén varia. Siempre se apunta q la cabecera de la pila.

El procesador se determina q ejecuta llamadas instrucciones de máquina.

Al conjunto de instrucciones q puede ejecutar el procesador se llama repertoire de instrucciones del procesador.

• Puntero: dirección base de la pila. El puntero decreciente o incremento q si se crece o se reduce un elemento.

Bucle: contiene la dirección base del bloque reservado p/ la pila. Si se intenta un pop q la pila vacía, crece un error.

Límite de pila: dirección del otro extremo del bloque reservado. Si se intenta un push q la pila llena, da error.

Organización registros CPU Motorola 68000	
	32
	80
A0	D1
A1	D2 Propósito gral.
A2	D3 p/ cualquier dato
A3	D4
A4	D5 p de 32 bits
A5	D6
A6	D7
A7 Apuntador de start word	
Pc o valor E	superior A7' 32 bits PC : 8 bits resto de bits

• 8 reg de 32 bits de datos

• 9 reg de direcciones

• 2 stacks, 1 p/ usuar., 1 p/ S.O.

base de datos de 32 bits de memoria

Instrucciones - Forma: destinatario destino, fuente. Destino y fuente son 2 operando.

Intel
donde 1 de ellos está especificado x alguno de los modos visto, el otro operando es un reg de la CPU

Llamado: men: especificar de una dirección de memoria

reg: reg de la CPU

mem: dato inmediato.

1 operando se referencia usando un modo de direccionamiento

• Instrucción mem, reg : " reg, reg " men, imm " entre parentesis "

• Inst. reg, mem : " reg, imm "

• El nombre destino y fuente proviene del hecho q si hay un mno de datos, es decir los otros(fuente) tienen q seguir la siguiente(destino)

En una suma hoy 2 operandos y el res. se almacena en el lugar del operando segundo (destino)

DATA

• No hace direccionamiento x memoria

Instrucciones - Intel 8086 • ADD AX,BX → AX = AX+BX • MOV AL,CH → AL=CH

• ADD AL, AH → AL = AL+AH • SUB AX,BX → AX = AX-BX

• Direccionamiento por registro

• 16 bits → 16 bits → presenta entredos igual

• ADD AX,35AFh → AX = AX+35AFh • MOV AL,3EH → AL = 3EH 8 bits

modo directo • ADD AL,JS+AL = AL+JS • SUB AX,J234h → AX = AX-J234h

• Direccionamiento inmediato

gratuito de la celda

• ADD AX,[35AFh] → AX = AX+ contenido directo 35AFh , 3500h

• ADD AL,DATA → AL = AL + contenido variable DATA (8 bits)

• MOV CH,NUM1 → CH = contenido variable NUM1(8 bits)

• Direccionamiento directo

• Direc + bajo : significativa

• 8 bits

• ADD [BX],BX → AX = AX+ dato almacenado en dirección contenida en BX y lo que sigue → p/ completar 16 bits.

• MOV [BX],AL → dato almacenado en BX = AL

• Direccionamiento por registro(indirecto)

• MOV CX,[BX+SI] → CX = dato almacenado en la direc BX+SI y lo que sigue

Ejemplo • MOV EBX+DI],AL → Dato almacenado en BX+DI = AL

• Direccionamiento base + indice

• MOV AL,[BX+2] → AL = dato almacenado en dir BX+2

• MOV [BX+2Ah],AX → dato almacenado en dir BX+2Ah y lo q sigue = AX (16 bits)

• Direccionamiento relativo por registro

• MOV AL,[BX+SI+2] → AL = dato almacenado en la dir BX+SI+2

• MOV [BX+SI+2Ah],AX → dato al " y lo q sigue = AX (16 bits)

• Direccionamiento relativo base + indice

- Formato de instrucción - ordenes al disco:
 - Instrucción carta o bájos dependencia bits
 - N° bits/fog → ancho de banda de memoria de orden
 - Velocidad procesador/velocidad memoria
 - Instrucción + carta "parce + rápido" el procesador
- Suficientes bits p/ expresar todas las op deseadas.
- La esp. dentro de cada bits libres p/ el futuro.
- Cada dígitos de orden

El empleo de la fuerza de atracción entre los átomos y las moléculas es el factor que determina la viscosidad de un líquido.

- Editar prueba.com - Usar ed /x de folio
 - Ensamblar prueba.com - Usar Antes (prueba.o + prueba.lib)
 - Enlazar prueba.o - Usar lNK86 + prueba.eje
 - Usar MSVCR80.dll → carga prueba.eje y ejecutar

ORG 2000H
MOV BX,3000H
MOV AX,[BX]
ADD BX,FO2H

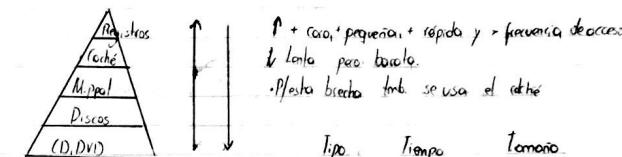
Símbolos
nombre: tipo: valor

Clase 9 → Subsistemas de memoria / dg. mem. par.

Memoria: vel. procesador: duplica el 1/18 meses las inst. ejecutadas /seg.
memoria: cuaduplica tamaño el 1/36 meses.

Este genera desequilibrio el procesador y memoria. P/ equilibrar esto biehn se usan <> tipos de memoria desde rapido a lento (registros) y lento a lento (discos). Su interac se aprovecha p/ un comportamiento equivalente al de una memoria unica, rapido y grande (la cual sea considerada ideal).

○ Jerarquía de memorias. Forma en q se organizan los \Rightarrow tipos de memoria



Memoria del consejero de

- Tensión \leftrightarrow
 - Fundamentos físicos \leftrightarrow
 - Localización \leftrightarrow

} Se unen p/
lo capaz de almacenar
fuerza de arrastre resistido.

Características

Drox de la info:

- Volatile RAM
 - No volatile discs, instead
 - Permanent: ROM, EEPROM

Modo de direccionamiento

Acceso x palabra: mem 200

* bloqué: discus, rock

Velocidad

• Monaria semirond

Tiempo acceso tiempo/
inicio de op hasta obtención D

relocation
instructions

Memoria magnetica
ocesso: passo x cobrindo

V. transference, 6 fm/sec

Métodos de acceso

- Alatario: tiempo p/ acceder a una linea dado es independiente de la cantidad de direcciones o direcciones y es constante p/ memoria.
- Semicontinuas o sencillas o lateral o superficie.
- Directo: la bloquea y registra tiene una dirección q se basa en la linea física.
- Secuencial: memoria cache

Memoria de acceso aleatorio

RAM (Random Access Memory). Aleatorio significa q se puede acceder a cualquier celda de memoria en el mismo tiempo, independiente de la posic en la estructura de memoria.

- SRAM: memoria estatica basada en flip-flops.

- DRAM: dinamico basada en transistores (capacitores) (celdas q tienen carga 0 con carga electrica).

- ROM: mismo tipo de acceso.

DRAM: memoria + info q SRAM en la misma superficie. Represa, lento, lento.

Hoy q "refrescar" p/ memoria p/ q SRAM se refresca x tendencia natural a desarreglar. SRAM + rapido, usado como cache.

Organizar el chip basicamente de una memoria de semiconductores es lo chip de memoria.

Todos los cellos de memoria de semiconductores comparten 3 propiedades:

- Bistable: representa 1 y 0.

- Se puede escribir en ellas al menos 1 vez.

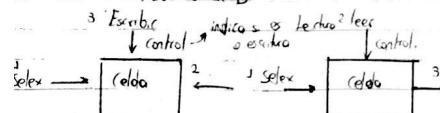
- Se pueden leer p/ conocer el E.

El chip tiene 3 terminales: p/ tener una p/ q ser memoria.

- Select: selecciona una celula de memoria.

- Control:

- E/Lectura de D



Escritura de D

Lectura de D



Organizar de la memoria

• Una memoria de 1 bit lo implementamos q flip-flops y "armamos" seguidos. de n bits dff.

• P/ construir memorias "grandes" se requiere una qg >, en la cual sea posible discriminar p/ obtener individuales

Organizar del chip

• El chip contiene un arreglo de celulas de memoria.

• En las memorias de semiconductores se han empleado 2 tipos: 2D y 2½D

Organizar 2D

• El arreglo este organizado en 2º polobras de 8 bits q/ lineas horizontales (una de 2⁸) se conecta a el bus de memoria, se le da un renglon.

• Los buses verticales conectan el bus a los S.

• El bus de direcciones q esta en el chip tiene 2⁸ salidas p/ los 256 bits del bus de direcciones.

Organizar 2½D

• El arreglo es "cuadrado" y funciona = q= 20

• Los bits de una misma polobra estan dispersos en 2 chips.

• La direc se divide en 2 partes: 1 selec de renglon y una selec de columnas. Hoy 2 direc fijas.

Comparar

• En 2D todos los bits estan en el mismo chip.

• En 2½D los bits de una misma polobra estan en 2 chips.

• 2D es lento, no grande de polobras de pocos bits. El linea de selección de polobras tiene q tener un multiplexor y conectarlo al decodificadores. Quedan mucha superficie.

• 2D difiere el uso qfiz de los circuitos correctores de errores. El 2½D q tener los bits dispersos en 2 chips hoy < probabilidad de error.

NOTA

En 2560 la fila y columna indican la dirección del los datos frescos.

Problemas: 1) /módulo tiene el espacio de direccionamiento requerido, pero solo cubre una parte de la palabra.

Sol: varios módulos en paralelo.

2) La longitud de la palabra es lo deseado, pero los módulos no tienen la capacidad deseada.

Sol: cubrir un campo de direc. c/ módulos de memoria "en serie".

1 módulo estaria en directo \leftrightarrow

Nuevas técnicas RAM

• La DRAM es la misma desde los 1º chips de RAM.

• Enhanced DRAM (tamaño pequeño SRAM)

La SRAM guarda lo ultimo leído hasta la fecha (como una cache).

• Cache DRAM (cache una SRAM + grande)

Se usa la SRAM como cache o como buffer serial.

• Synchronous DRAM (SDRAM) p/ compensar la latencia de RAM

Adelant en DRAM. Acceso sincronizado al reloj externo. Se presenta una directa a la RAM. RAM encierra D y no ejercita la DRAM.

SDRAM maneja D neto en tiempo del reloj.

CRU puede hacer otra cosa mientras opera.

Burst permite al SDRAM tratarlos en bloques.

Capa 4: memoria interna.

4 = directa

3H = bus de direc.

3H → bus de reg.

2L = bus de datos

ATA
SATA

Práctica 5 - Lenguaje Assembly

Procedimientos:

• Interpretar 1 y 0

• Resetea op. elementos como suma, resta, etc.

• Copiar octalode p/ el tamaño de D

Uso de D: Dificil recordar el código binario p/ d/ instrucción

Requiere resolver op. complejas.

Usa D de tamaño variable y grande.

○ **Lenguaje assembly:** Aprende sintaxis recordando lo que "migra" mediante mnemónicos.

• Implementa E de control \rightarrow Q, lygas + saltos condicionales y E/S.

• Tamaño D de \rightarrow tamaños de direcciones

una palabra \leftrightarrow 4 bytes

Instrucciones: codigos binarios q' dicen q' hace el q' tiene sus inst.

q' inst \leftrightarrow representación binaria (muchas inst. cortas + largas)

q' modo de direccionamiento \rightarrow + modos = códigos de inst + largos.

tmb. es una inst \leftrightarrow

Sx88 \rightarrow p/lo practica \rightarrow 4 registros: de 32 bits: Ax, Bx, Cx, Dx

A su vez pueden dividirse en 8 bits: AH, AL, BH, BL, etc.

Tamaño de inst. variable (multiplos de 8 bits)

de palabra de 32 bits

Bus de D de 8 bits de ancho (2 relojes p/ recuperar una palabra)

Direc. de 32 bits (2^{32} : 4GB de memoria)

Modo de d/c en memoria: Inmediato-Directo (registro)-Indir (memoria)-Ind x reg

Tener en cto: • linea ENTRADA final del archivo si o si

• Nombre de archivo máximo 8 caracteres

Estructura de un programa 1º instrucciones basadas en directo programando (2000 h)

Las d se dividen a partir de la direc. 2000h

Otra instr. a partir de q direct. se dividen las sig. instr. y declaraciones

Las d e inst. declarativas en cualquier posic., siempre q ubicamos en 2000h la 1º inst. del programa.

Flags: 8 banderas en total

Solo afectado x op. aritméticas y op. lógicas de la ALU

Al resto de op. no afectan a los flags

Cuando altera los flags tras restar/líz operando pero no almacenarlo de nuevo

de ninguno de ellos.

Señal importantes el Z, S, O, C.

Transferencia de control Sig. inst. alteran el flujo normal del programa modificando el sig. IP o ↗ condicionales

Instancia cond. x operador

JZ F Z=true JNZ

JS F S,ng=true JNS

JO F overflow=true JNO

JC Flag carry=true JNC

JUP Trasn. normal No opera

CUV 5x8B. here su propio ensamblador ASME8.

Planteamiento del programa, teclado MS8B <ca>

Instrucciones

- Inst. transf. de D: NOP

- operaciones lógicas: Inst. entretejidas ADD, ADC, SUB, SBB

- lógicas AND, OR, XOR, NEG, NOT

- comparación CMP

- incremento / decremento INC, DEC

- cambios de flags de programa Sellos condicionales JZ, JNZ, JS, JNS, JC, JNC, JO, JNO

- Sellos condicionales JMP

- salida a pantalla CALL, RET

- mueve de pila PUSH, POP, PUSHF, POPF

- " gestión de los interrupciones: INT, IRET, STI, CLI

- de control NOP, HLT

- ES: IN, OUT

W: ancho tamaño de operandos W: 0 → 8 bits DB → 1 byte / 8 bits

W: 1 → 16 bits DW → word → 2 bytes

ID = dirección de la próxima instr. a ejecutar

SP = dirección del topo de la pila

Instrucciones de salto

JZ salto si el flag Z=1 JNZ salto si Z=0, ↗ rega

JS " S:1 JNS " :D:1 ↗ rega

JC " C:1 JNC " C:0, ↗ rega

JO " O:1 JNO " O:0, ↗ rega

JMP " dirección ; salto siempre

Selección if then else

JF AL=4 then then MOV BL,1

begin JZ then INC CL

BL:1; JMP Else

CL=CLs; JMP Fin_IF

end Else: MOV BL,2

DFC CL

Fin - JF: HLT