CONCEPTOS DE ARQUITECTURA DE COMPUTACION

Lenguaje Maquina: lenguaje binario en el que se definen y almacenan las instrucciones en memoria

Lenguaje Ensamblador: lenguaje simbolico que reemplaza los codigos de operacion y direcciones del lenguaje maquina

Formato de Instruccion - Campos:

1.- Codigo de Operacion - Opcode
2.- Direccion /es
3.- Modo

Ciclo de Operacion Basico:

1.- Obtener la instrucion de memoria. Almacenarla en registro de control
2.- Decodificar la instruccion
3.- Localizar los operandos empleados en la instruccion
4.- Obtener de la memoria los operandos (si fuese necesario)
5.- Ejecutar la operacion en la ruta de datos
6.- Almacenar el resultado en un lugar adecuado
7.- Volver a paso 1 y obtener la siguiente instruccion

Registro Importante: PC - contiene la direccion de la siguiente instruccion a ejecutar

CONJUNTO DE REGISTROS: registros a los que el programador tiene acceso

- Archivo de registros
- PC - Contador de programa
- SP - Puntero de Pila
- PSR - Estado del procesador

OTROS REGISTROS:

- IR - registro de instruccion
- Registros ocultos (transparentes al usuario)
- Registros de canalizacion (pipeline)
- CAR -

DIRECCIONAMIENTO OPERANDOS:

3 DIRECCIONES - ADD T1, A, B M[T1] <- M[A] + M[B]

2 DIRECCIONES - ADD T1, B M[T1] <- M[T1] + M[B]

1 DIRECCION - ADD B ACC <- ACC + M[B]

0 DIRECCIONES - ADD Estructura de pila LIFO

(A+B) * (C+D)

3 DIRECCIONES:

ADD T1, A, B	M[T1] <- M[A] + M[B]
ADD T2, C, D	M[T2] <- M[C] + M[D]
MUL X, T1, T2	M[X] <- M[T1] * M[T2]

3 DIRECCIONES:

ADD R1, A, B	R1<- M[A] + M[B]
ADD R2, C, D	R2 <- M[C] + M[D]
MUL X, R1, R2	M[X] <- R1 * R2

2 DIRECCIONES:

MOVE T1, A	M[T1] <- M[A]
ADD T1, B	M[T1] <- M[T1] + M[B]
MOVE X, C	M[X] <- M[C]
ADD X, D	M[X] <- M[X] + M[D]
MUL X, T1	M[X] <- M[X] * M[T1]

1 DIRECCION:

LD A	ACC <- M[A]
ADD B	ACC <- ZCC + M[B]
ST X	M[X] <- ACC
LD C	ACC <- M[C]
ADD D	ACC <- ACC + M[D]
MUL X	ACC <- ACC * M[X]
ST X	M[X] <- ACC

0 DIRECCIONES:

PUSH A	TOS <- M[A]
PUSH B	TOS <- M[B]
ADD	TOS <- TOS + TOS_{-1}
PUSH C	TOS <- M[C]
PUSH D	TOS <- M[D]
ADD	TOS <- TOS + TOS_{-1}
MUL	TOS <- TOS * TOS_{-1}
POP X	M[X] <- TOS

ARQUITECTURAS DE DIRECCIONAMIENTO:

1.- Memoria-Memoria (1-3 operandos)

2.- Registro-Registro o de carga/almacenamiento de 3 direcciones (1 sola a memoria)

3.- Registro-memoria (3, 2 operandos - 2, 1 memoria)

4.- de Acumulador Sencillo (1 operando)

5.- Arquitectura de Pila (0 operandos)

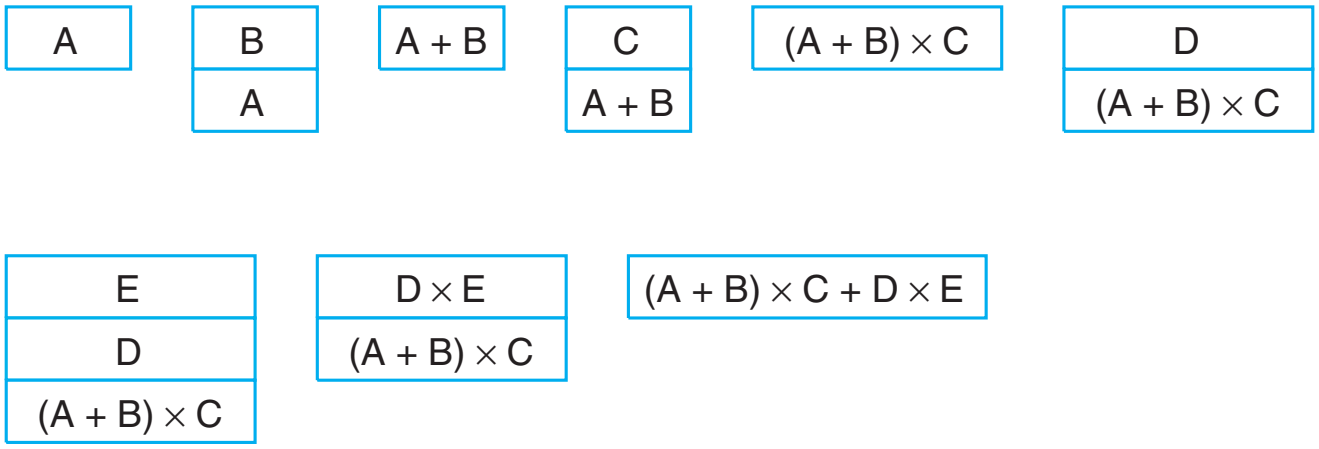


Fig. 4.1 Stack Activity for Execution of Example Stack Program

Fig. 4-2 Instruction Format with Mode Field

MODOS DE DIRECCIONAMIENTO:

Distintos Modos de Direccionamiento:

- Flexibilidad a la programacion
- Reducen cantidad bits campos de direccion de la instruccion

Direccion Efectiva - Direccion donde se encuentra el operando

---

- Modo Implicito
- Modo Inmediato
- Modos de Registro y de Registro Indirecto
  - Modo Registro
  - Modo Registro Indirecto
  - Modo Autoincremento / Autodecremento
- Modo de direccionamiento directo
- Modo de direccionamiento indirecto
- Modo de direccionamiento relativo
- Modo de direccionamiento indexado o indizado

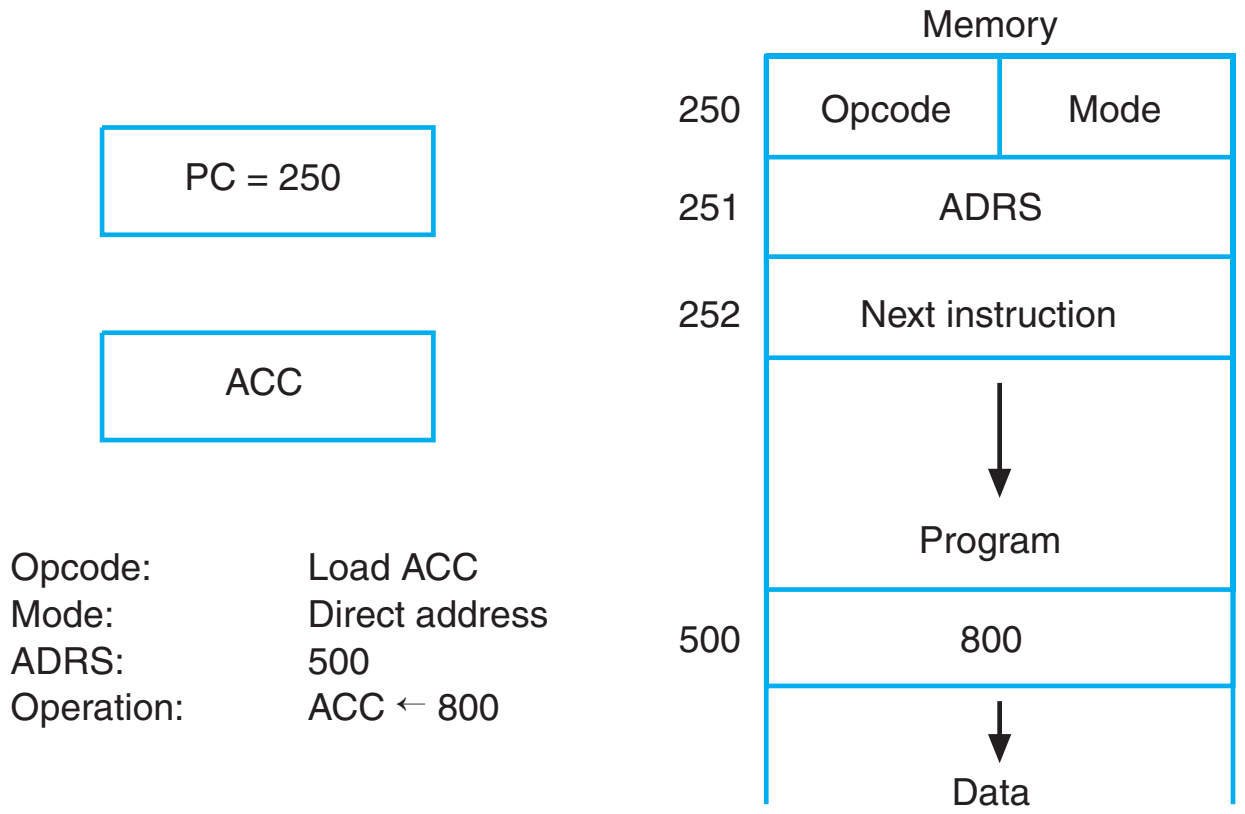


Fig. 4- 3 Example Demonstrating Direct Addressing for a Data Transfer Instruction

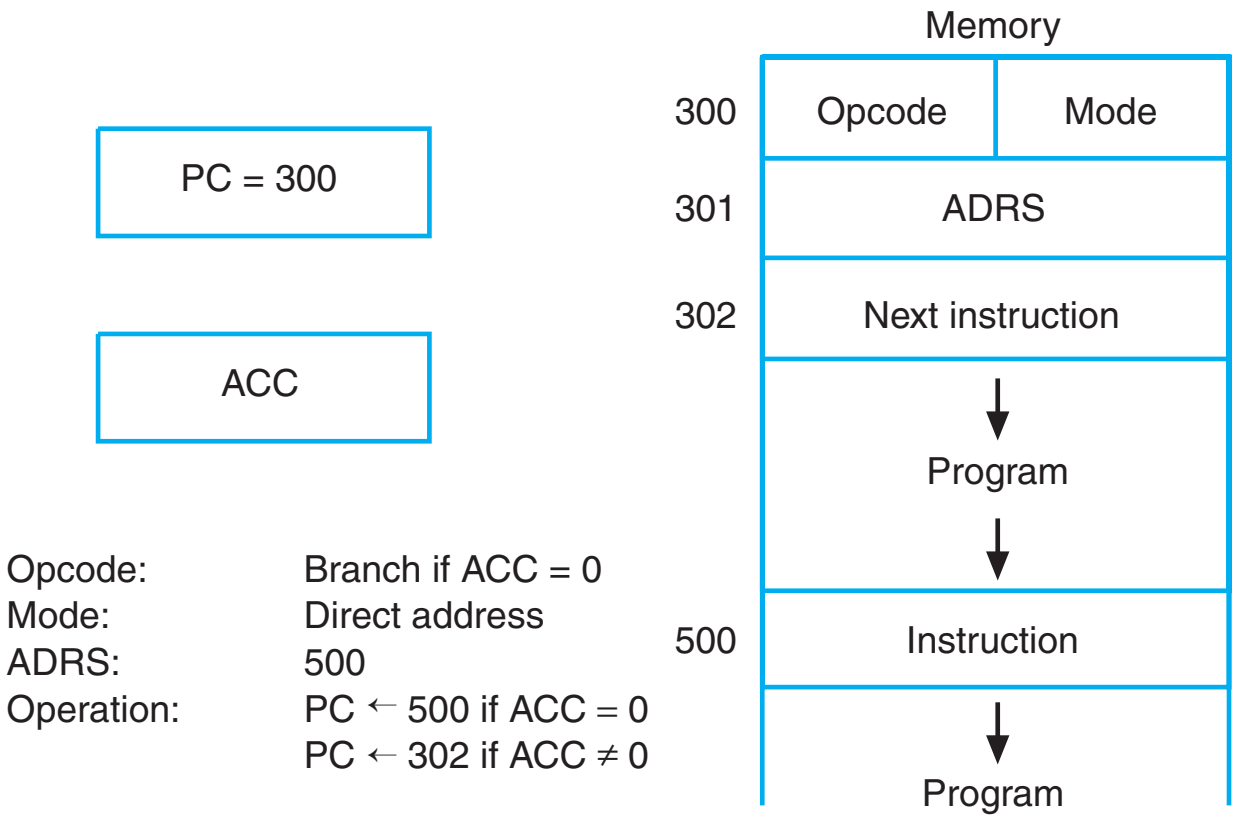


Fig. 4 - 4 Example Demonstrating Direct Addressing in a Branch Instruction

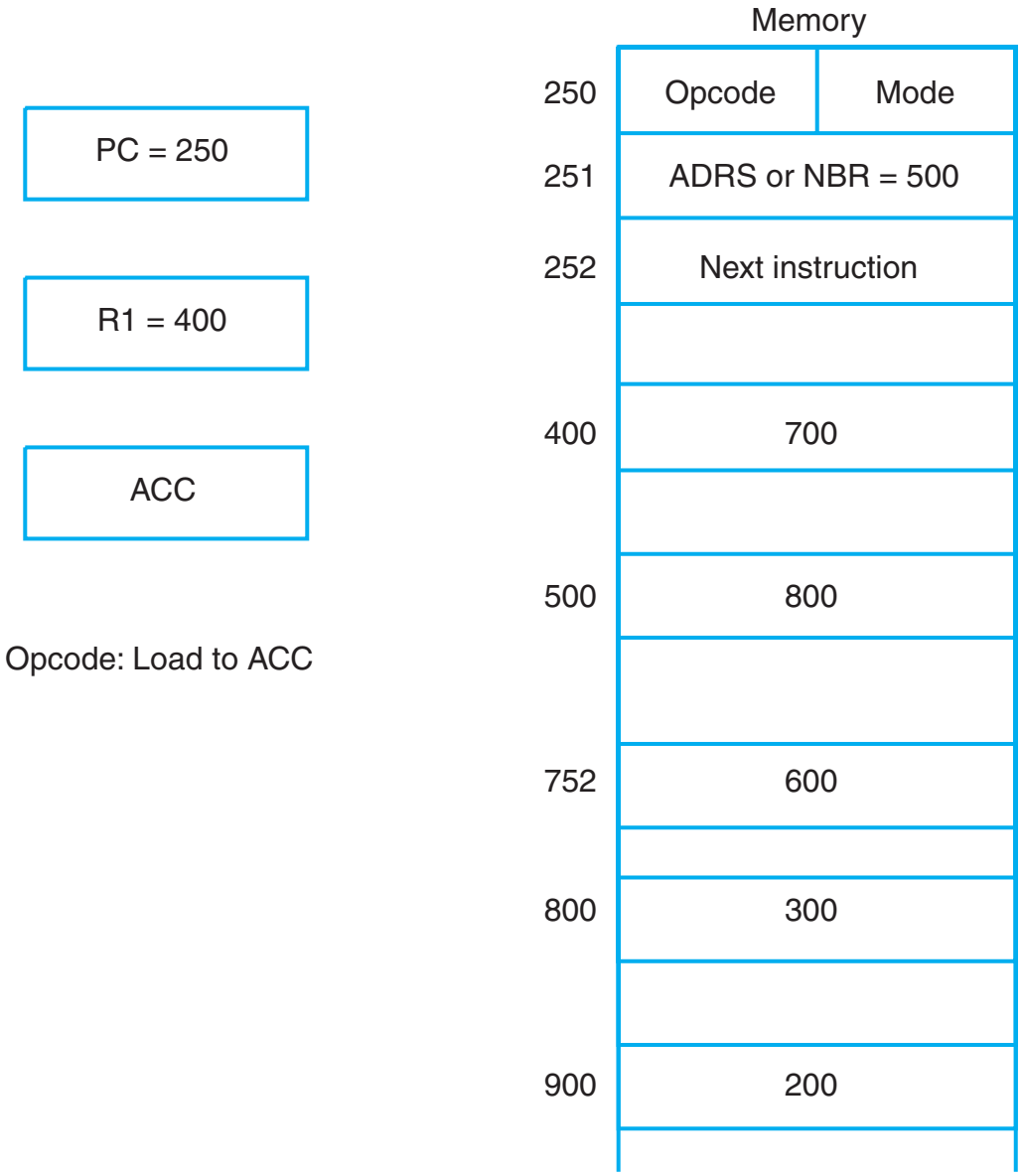


Fig. 4-5 Numerical Example for Addressing Modes

TABLE 4-1
Symbolic Convention for Addressing Modes

			Refers to Figure 5-4	
Addressing mode	**Symbolic convention**	**Register transfer**	**Effective address**	**Contents of *ACC***
Direct	LDA ADRS	$ACC \leftarrow M[ADRS]$	500	800
Immediate	LDA #NBR	$ACC \leftarrow NBR$	251	500
Indirect	LDA [ADRS]	$ACC \leftarrow M[M[ADRS]]$	800	300
Relative	LDA $ADRS	$ACC \leftarrow M[ADRS + PC]$	752	600
Index	LDA ADRS (R1)	$ACC \leftarrow M[ADRS + R1]$	900	200
Register	LDA R1	$ACC \leftarrow R1$	—	400
Register indirect	LDA (R1)	$ACC \leftarrow M[R1]$	400	700

Table 4-1 Symbolic Convention for Addressing Modes

ARQUITECTURAS DE CONJUNTO DE INSTRUCCIONES

RISC
Reduced Instruction Set Computers

1.- Accesos a memoria restringidos a carga y almacenamiento

2.- Archivo de registros grande

3.- Formatos de Instruccion de la misma longitud

4.- Cantidad de modos de direccionamiento limitada

5.- Instrucciones ejecutan operaciones elementales

META: Instrucciones rapidas

Unidad de control sencilla y cableada

Diseæo en canalizacion

CISC
Complex Instruction Set Computers

1.- Acceso a memoria disponible en la mayoria de instrucciones

2.- Archivo de registros pequeæo

3.- Formatos de Instruccion de diferente longitud

4.- Cantidad de modos de direccionamiento sustancial

5.- Instrucciones ejecutan tanto operaciones elementales como complejas

META: Programas compactos - Ahorro de memoria

Unidad de Control Compleja - Microprogramada

Puede incorporar canalizacion

OPERACIONES ELEMENTALES

1- Transferencia de datos — No cambian el dato

2- Manipulacion de datos — Transfieren el dato procesado

- Aritmeticas
- Logicas
- Desplazamiento

3- De control de programa — Cambian la secuencia de ejecucion de las instrucciones. Para ello cambian el contenido del PC

TABLE 4-2
Typical Data Transfer Instructions

Name	Mnemonic	
Load	LD	registro <- memoria
Store	ST	memoria <- registro
Move	MOVE	
Exchange	XCH	
Push	PUSH	operaciones de pila
Pop	POP	
Input	IN	registros <-> puertos
Output	OUT	

Table 4-2 Typical Data Transfer Instructions

VENTAJA: procesador se refiere a la pila en modo implicito puesto que SP siempre apunta al TOS

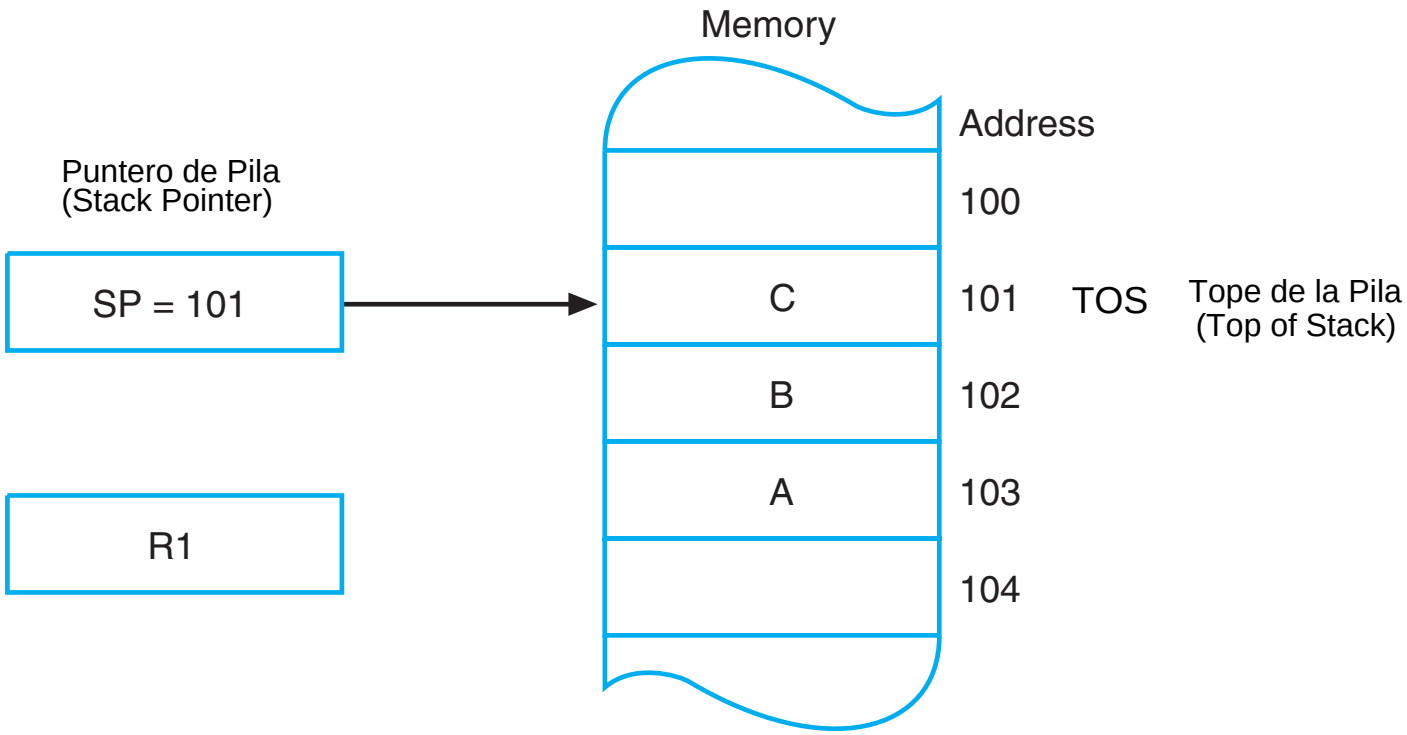


Fig. 5 - 6 Memory Stack

POP: R1 <- M[SP]
SP <- SP + 1

PUSH: SP <- SP - 1
M[SP] <- R1

Variaciones: pila que crece aumentando direccion.
SP apunta a la primera direccion vacia en vez de al TOS

INSTRUCCIONES DE ENTRADA/SALIDA (Input/Output) (I/O)

Son operaciones de transferencia entre el archivo de registros del procesador y los dispositivos de entrada/salida

Puerto: Registro con lineas de I/O conectadas a un dipositivo de entrada/salida

El puerto se selecciona mediante una direccion que se incluye en las instrucciones de entrada/salida

Dos modos de asignacion de direcciones de puerto:

- Sistema de E/S independiente o configuracion de E/S aislada (Input/Output)
- Sistema de E/S correlacionada con memoria (Load/Store)

TABLE 4-3
Typical Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Subtract reverse	SUBR
Negate	NEG

Table 4-3 Typical Arithmetic Instructions

TABLE 4-4
Typical Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Set	SET
Complement	NOT
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC

Table 4-4 Typical Logical and Bit Manipulation Instructions

TABLE 4-5
Typical Shift Instructions

Name	Mnemonic		
Logical shift right	SHR	aæade 0	
Logical shift left	SHL	no cambia V	aæade 0
Arithmetic shift right	SHRA	mantiene bit de signo	
Arithmetic shift left	SHLA	cambia V	aæade 0
Rotate right	ROR		
Rotate left	ROL		
Rotate right with carry	RORC		
Rotate left with carry	ROLC		

Table 4-5 Typical Shift Instructions

Desplazamiento Multiple:

OP cod. op.	REG reg, dir	TYPE log, arit	RL der, izq	COUNT num pos

ARITMETICA DE PUNTO FLOTANTE

Dos partes:

signo + fraccion (mantisa)
exponente

Ejemplo base 10:

Fraccion	Exponente	
+.61327	+04	=> $0.61327*10^{4} = 6132.7$

Ejemplo base 2:

Fraccion	Exponente	
01001110	000100	=> $(0.1001110)*2^{4}$

Indica signo positivo

NUMERO NORMALIZADO: si el digito mas significativo de la fraccion es distinto de 0

Ejemplo base 10:

0.350	normalizado
0.00427	no normalizado

Representacion de punto flotante aumenta la gama de numeros que se pueden representar con n bits a expensas de perder precision

Ejemplo: con 48 bits

complemento a dos: $-2^{47} \leq Z < 2^{47}$ aprox +/- 10^{14}

punto flotante: 1 signo + 35 fraccion + 12 exponente

+/- $(1-2^{-35})x2^{2047}$ aprox +/- 10^{615}

SUMA / RESTA:

Requiere que los puntos base esten alineados. Las partes exponeneciales deben ser iguales

Se desplaza el sumando de exponente menor tantos lugares a la derecha como la diferencia entre los exponentes

EJEMPLO (base 10):

Suma:

$$\begin{array}{r} .5372400 \times 10^{2} \\ +\ .1580000 \times 10^{-1} \\ \hline \end{array} \qquad \begin{array}{r} .5372400 \times 10^{2} \\ +\ .0001580 \times 10^{2} \\ \hline .5373980 \times 10^{2} \end{array}$$

Resta:

$$\begin{array}{r} .56780 \times 10^{5} \\ -\ .56430 \times 10^{5} \\ \hline .00350 \times 10^{5} \end{array}$$

MULTIPLICACION / DIVISION:

Multiplicacion: se multiplican las fracciones y se suman los exponentes

Division: se dividen las fracciones y se suman los exponentes

---

La mayor parte de los ordenadores efectuan un procedimiento de normalizacion tras cada operacion, para asegurar que los resultados estan normalizados

EXPONENTE SESGADO:

SESGO: numero que se suma al exponente de modo que todos los exponentes se vuelvan positivos.

Ejemplo (base 10): rango exponentes E (-99, +99).
Utilizando sesgo 99 e=E+99 => e (0, +198)

Ejemplo (complemento a dos): n bits. Sesgo $2^{n-1}-1$

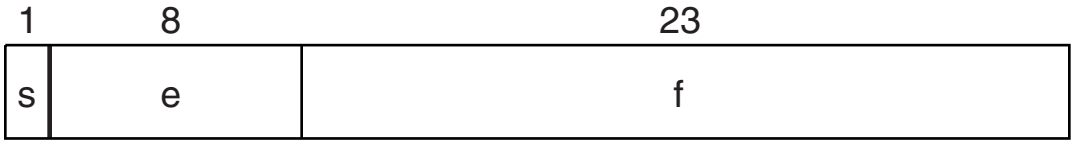
VENTAJA: Las comparaciones de los exponentes son mas sencillas

FORMATO DE OPERANDO STANDARD:

2 formatos - precision sencilla - 32 bits
- doble precision - 64 bits

Sufijo F - indica operacion punto flotante
Sufijo FS - precision sencilla
Sufijo FL - doble precision

PRECISION SENCILLA - FORMATO STANDARD IEEE:



exponente sesgado (sesgo 127)

1. implicito en la fraccion

Fig. 4-7 IEEE Floating-Point Operand Format

SIGNIFICANDO: 1.f siempre normalizado (1.00...0, 1.11...1)

Ejemplos:	Campo f	Significando	Equivalente decimal
	100...0	1.100...0	1.50
	010...0	1.010...0	1.25
	000...0	1.000...0 (*)	1.00 (*)

(*) suponiendo que el exponente no sea igual a 00...0

EXPONENTE: E[-126, +127]

e[00000001, 11111110]

NUMEROS NORMALIZADOS:

$$(-1)^{S} \times 2^{e-127} \times (1.f)$$

num positivo mayor a representar: f = $1+1-2^{-23} = 2 - 2^{-23}$; e = 11111110, E = 254 - 127 = 127 | $(2-2^{-23})*2^{127}$

num positivo menor a representar: f = 1; e = 00000001, E = 1-127 = -126 | 2^{-126}

TABLE 4-6
Evaluating Biased Exponents

Exponent *E* in decimal	Biased exponent $e = E + 127$	
	Decimal	**Binary**
− 126	− 126 + 127 = 1	00000001
− 001	− 001 + 127 = 126	01111110
000	000 + 127 = 127	01111111
+ 001	001 + 127 = 128	10000000
+ 126	126 + 127 = 253	11111101
+ 127	127 + 127 = 254	11111110

Table 4-6 Evaluating Biased Exponents

Exponente Sesgado $e = E + 127$			
Decimal	***s***	**Exponente e**	f
+ ∞	0	11111111	0
- ∞	1	11111111	0
NaN	0o1	11111111	≠0
0	0o1	00000000	0
desnorm		00000000	≠0

INSTRUCCION DE CONTROL DE PROGRAMA: es aquella que al ejecutarse puede cambiar la direccion del PC y causar la alteracion de la secuencia de instrucciones del programa

Bits Estado	Significado
Z	resultado 0
C	acarreo
N	signo resultado
V	rebasamiento

TABLE 4-7
Typical Program Control Instructions

Name	Mnemonic	
Branch	BR	Condicionales o incondicionales
Jump	JMP	
Skip next instruction	SKP	
Call procedure	CALL	Llamada subrutina y retorno
Return from procedure	RET	
Compare (by subtraction)	CMP	3 tipos
Test (by ANDing)	TEST	

relativo
directo o indirecto

no retiene resultado
no cambia bits estado (tipos 1 y 2)

Table 4-7 Typical Program Control Instructions

Tipos CMP y TEST:

- Tipo 1 - Bifurcacion
- Tipo 2 - Cambia contenido de un registro
- Tipo 3 - Cambia bits de estado

TABLE 4-8
Conditional Branch Instructions Relating to Status Bits in the PSR

Branch condition	Mnemonic	Test condition
Branch if zero	BZ	$Z = 1$
Branch if not zero	BNZ	$Z = 0$
Branch if carry	BC	$C = 1$
Branch if no carry	BNC	$C = 0$
Branch if minus	BN	$N = 1$
Branch if plus	BNN	$N = 0$
Branch if overflow	BV	$V = 1$
Branch if no overflow	BNV	$V = 0$

Table 4-8 Conditional Branch Instructions Relating to Status Bits in the PSR

NOTA IMPORTANTE:

En la mayoria de las maquinas el bit C es acarreo (carry) en operaciones de suma y prestamo (borrow) en las de resta
Es decir C=CO sumador en suma y C=$\overline{CO}$ sumador en resta

C en operaciones de desplazamiento a la izquierda tambien sirve para inspeccionar el valor del bit que sale

TABLE 4-9
Conditional Branch Instructions for Unsigned Numbers A-B

Branch condition	Mnemonic	Condition	Status bits*
Branch if higher	BH	$A > B$	$C + Z = 0$
Branch if higher or equal	BHE	$A \geq B$	$C = 0$
Branch if lower	BL	$A < B$	$C = 1$
Branch if lower or equal	BLE	$A \leq B$	$C + Z = 1$
Branch if equal	BE	$A = B$	$Z = 1$
Branch if not equal	BNE	$A \neq B$	$Z = 0$

*Note that *C* here is a borrow bit.

Table 4-9 Conditional Branch Instructions for Unsigned Numbers

TABLE 4-10
Conditional Branch Instructions for Signed Numbers A-B

Branch condition	Mnemonic	Condition	Status bits
Branch if greater	BG	$A > B$	$(N \oplus V) + Z = 0$
Branch if greater or equal	BGE	$A \geq B$	$N \oplus V = 0$
Branch if less	BL	$A < B$	$N \oplus V = 1$
Branch if less or equal	BLE	$A \leq B$	$(N \oplus V) + Z = 1$

Table 4-10 Conditional Branch Instructions for Signed Numbers

BE, BNE iguales que las intrucciones de bifurcacion para numeros sin signo

INSTRUCCIONES DE LLAMADA Y RETORNO DE PROCEDIMIENTO

PROCEDIMIENTO o SUBRUTINA:

- Secuencia autocontenida de instrucciones que ejecutan una tarea de computacion determinada
- Puede llamarse a una subrutina varias veces en diferentes puntos de un programa.
- Al llamar a una subrutina se bifurca el PC al inicio de esta.
  Tras su ejecucion se hace otra bifurcacion para regresar al programa principal.

LLAMADA A SUBRUTINA (CALL): (1 campo de direccion)

1.- Almacena valor de PC (direccion siguiente a la de llamada - Direccion de Retorno) en una localidad temporal (*)

2.- Carga en el PC la direccion especificada en la instruccion CALL que es la primera de la subrutina

REGRESO PROGRAMA LLAMADOR (RETURN): (sin campos de direccion)

1.- Carga el PC con la Direccion de Retorno

(*) - direccion de memoria fija, registro de procesador o PILA DE MEMORIA

CALL (pila):

SP <- SP-1	Disminuye apuntador de Pila PUSH
M[SP] <- PC	Almacena Direccion Retorno en la Pila
PC<- Dir Effect	Carga el PC con la primera inst subrutina

RETURN (Pila):

PC <- M[SP]	Transfiere Direccion Retorno a PC
SP <- SP+1	Incrementa Apuntador Pila. POP

INTERRUPCIONES

- Maneja una variedad de situaciones que requieren la suspension de la secuencia normal del programa
- Transfiere el control del programa en ejecucion a otro programa de servicio como resultado de una solicitud

INTERRUPCION v.s. LLAMADA PROCEDIMIENTO

1- Interrupcion se inicia en un punto impredecible del programa, no proviene de una instruccion

2- La direccion del programa de servicio se determina mediante un procedimiento de hardware (no por una instruccion)

3- Es necesario almacenar informacion que defina todo o parte del contenido del conjunto de registros (no solo el PC)

MODO USUARIO v.s. MODO SISTEMA - Modo determina bits del PSR (EI)

La mayor parte de las computadoras no responden a una interrupcion hasta que se complete la instruccion que se esta procesando

TIPOS DE INTERRUPCIONES

1- Externas - proceden de dispositivos E/S, de los de temporizacion, suministro de energia, etc..

2- Internas - surgen por el empleo ilegal o erroneo de una instruccion o una serie de datos - trampas

3- Software - es una instruccion especial de llamada que se comporta como interrupcion en lugar de llamada a subrutina
La puede utilizar el programador para iniciar un procedimiento de interrupcion en cualquier punto del programa

PROCESAMIENTO DE INTERRUPCIONES EXTERNAS

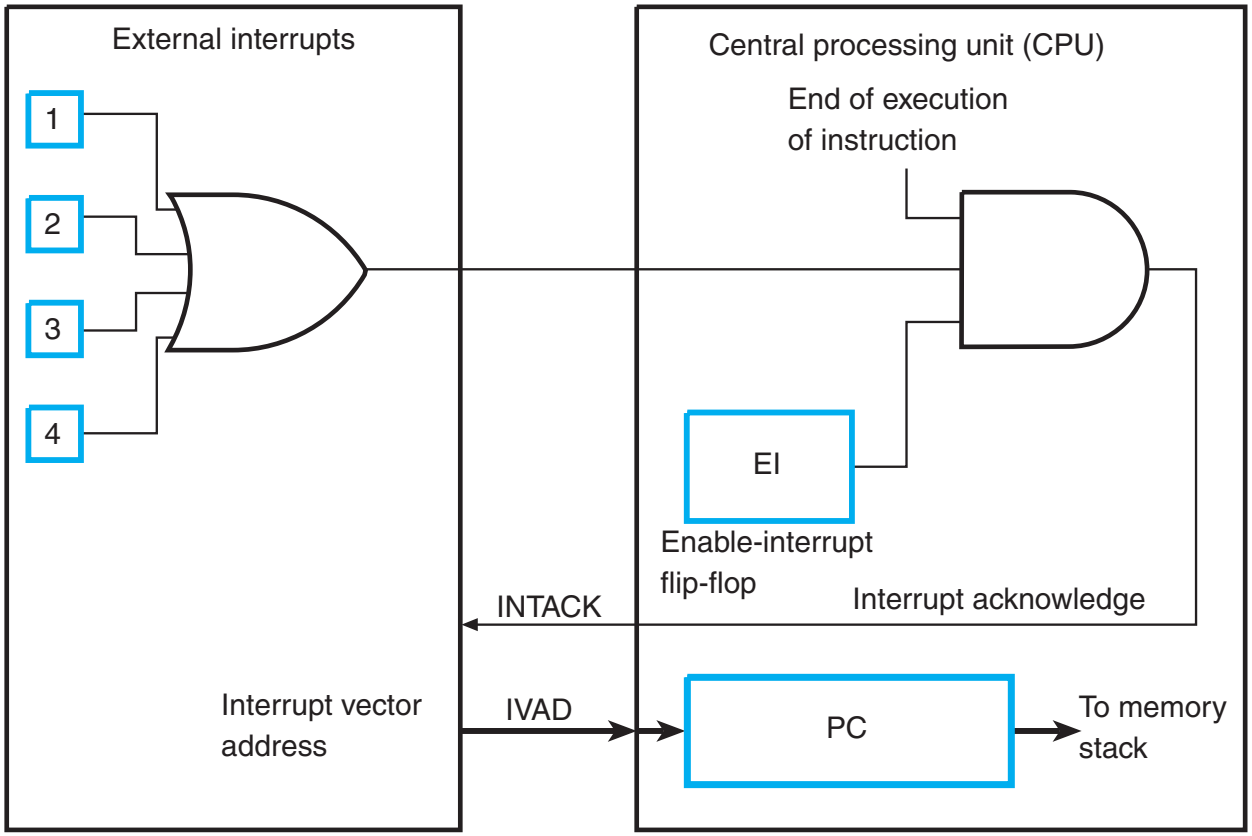


Fig. 4-8 Example of External Interrupt Configuration

Si EI=1, peticion interrupcion=1,
End of execution=1 => INTACK=1

IVAD - vector interrupcion

SERVICIO A INTERRUPCION

SP <- SP-1
M[SP] <- PC
SP<- SP-1
M[SP] <- PSR
EI <- 0
INTACK <- 1
PC <- IVAD

RETORNO PROGRAMA

- EI se restablece en su valor original
- Carga PC con direccion retorno
- POP en la pila

Instrucciones:

ENI - pone a 1 el flip-flop EI
DSI - pone a 0 el flip-flop EI