I.O. Teoricopràctic 3

Format d'instruccions i accessos a memòria

Què hem vist a teoria?

- Set d'Instruccions
  - ❑Tipus d'instruccions
  - ❑Diversitat de les instruccions
  - ❑Recursos requerits: Espai ocupat en memòria i temps d'execució

Què hem vist a teoria?

- En l'exemple estudiat en teoria:

Exemple:

•Analitzarem instruccions de diferent tipus i nombre d'adreces per computar l'expressió:

$$c= (a+b)*(a-b)$$

•Variables a,b,c i variable temporal x en posicions de memòria A, B, C i X, respectivament.
•Espai memòria: 16MB : 24 bits adreces
•BUS 32 bits

- Vam veure el següent:

Què hem vist a teoria

Instruccions amb accés a memòria i tres camps	Instruccions amb accés a memòria i dos camps	Instruccions amb accés a memòria + reg. acc	Instruccions amb load i Store + banc registres

Codi: (A,B,C, X pos. Mem)
ADD X, A, B
SUB C, A, B
MUL C, X, C
	Codi:
	MOV X,A
	ADD X, B
	MOV C, A
	SUB C, B
	MUL C, X
		Codi: (acc implícit)
		LOAD A
		ADD B
		STORE X
		LOAD A
		SUB B
		MUL X
		STORE C
			Codi: (Banc reg = 8)
			LOAD R1, A
			LOAD R2, B
			ADD R3, R1, R2
			SUB R4, R1, R2
			MUL R5, R3, R4
			STORE C, R5

Accessos a memòria - Bus dades = 32 bits - Accessos totals: 18	Accessos a memòria - Bus dades = 32 bits - Accessos totals: 23	Accessos a memòria - Bus de dades = 32 bits - Accessos totals: 14	Accessos a memòria - Bus de dades = 32 bits - Accessos totals: 12
Espai ocupat en memòria - Mida instr. 24 bits - Espai total = 36 Bytes	Espai ocupat en memòria - Mida instr. 56 bits - Espai total = 40 Bytes	Espai ocupat en memòria - Mida istr. 32 bits - Espai total = 28 Bytes	Espai ocupat en memòria - Mida instr. L/S: 35 bits - Mida instr. R: 17 bits - Espai total = 36 Bytes

Exercici 1

- Executa un codi similar al presentat en la taula anterior.
- Verifica l'espai que ocupa en memòria el programa
- Executa el programa en el simulador i compara el temps d'execució per al processador single stage i el pipeline de 5 etapes. Considereu que el clock per el primer processador i el del pipeline el podem conèixer a partir de la següent diapositiva

Exercici 1

Example 7.4 SINGLE-CYCLE PROCESSOR PERFORMANCE

Ben Bitdiddle is contemplating building the single-cycle MIPS processor in a 65 nm CMOS manufacturing process. He has determined that the logic elements have the delays given in Table 7.6. Help him compute the execution time for a program with 100 billion instructions.

Solution: According to Equation 7.3, the cycle time of the single-cycle processor is $T_{c1} = 30 + 2(250) + 150 + 200 + 25 + 20 = 925$ ps. We use the subscript "1" to distinguish it from subsequent processor designs. According to Equation 7.1, the total execution time is $T_1 = (100 \times 10^9 \text{ instructions})\ (1 \text{ cycle/instruction})\ (925 \times 10^{-12} \text{ s/cycle}) = 92.5$ seconds.

Table 7.6 Delays of circuit elements

Element	Parameter	Delay (ps)
register clk-to-Q	t_{pcq}	30
register setup	t_{setup}	20
multiplexer	t_{mux}	25
ALU	t_{ALU}	200
memory read	t_{mem}	250
register file read	t_{RFread}	150
register file setup	$t_{RFsetup}$	20

Single cycle processor time

$$T_c = t_{pcq\_PC} + 2t_{mem} + t_{RFread} + t_{ALU} + t_{mux} + t_{RFsetup} \quad (7.3)$$

Pipeline processor time

$$T_c = max \left\{ \begin{array}{c} t_{pcq} + t_{mem} + t_{setup} \\ 2(t_{RFread} + t_{mux} + t_{eq} + t_{AND} + T_{mux} + t_{setup}) \\ t_{pcq} + t_{mux} + t_{mux} + t_{ALU} + t_{setup} \\ t_{pcq} + t_{memwrite} + t_{setup} \\ 2(t_{pcq} + t_{mux} + t_{RFwrite}) \end{array} \right. \begin{array}{c} \text{Fetch} \\ \text{Decode} \\ \text{Execute} \\ \text{Memory} \\ \text{Writeback} \end{array} \quad (7.5)$$

comparador	t_{eq}	40 ps
AND	t_{AND}	15 ps
Memory write	$t_{memWrite}$	220 ps
Register write	$t_{RFwrite}$	100 ps

Modes d’adreçament de memòria

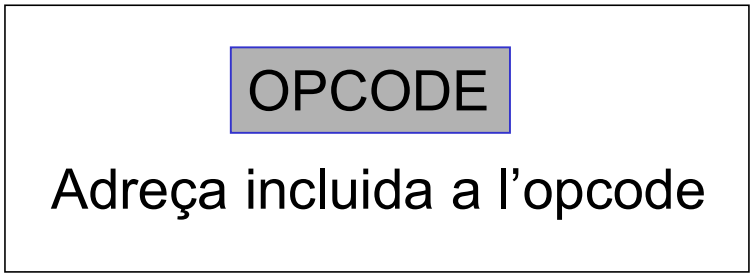
Tipus d’adreçament que tenim:

1. Mode d’adreçament implícit
2. Mode d’adreçament immediat
3. Mode d’adreçament directe
4. Mode d’adreçament indirecte
5. Mode d’adreçament relatiu
6. Mode d’adreçament Indexat

1. Mode d'adreçament implícit

•No necessita camp explícit d'adreça pq la ubicació de l'operand o el resultat ja està especificada a l'opcode.

•Pex. instruccions de clear del registre d'estatus o l'acumulador. registres són únics i implícits en l'opcode (aquests registres solen ser únics i estan per tant implícits en l'opcode)



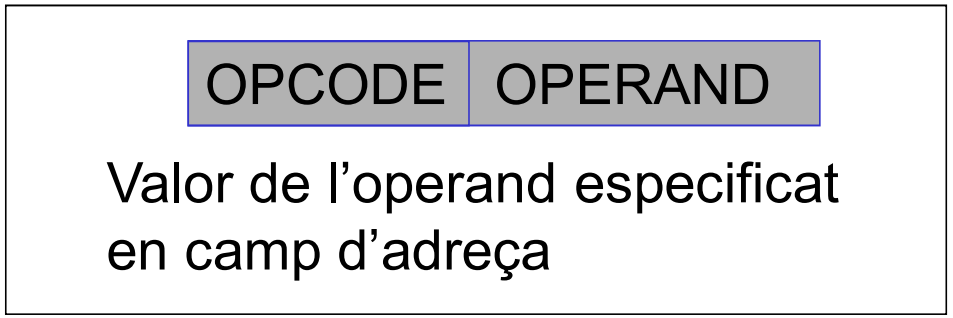
Adreçament Implícit

2. Mode d'adreçament immediat

•Operand especificat en camp d'adreça.
•Una instrucció d'adreçament immediat té un camp d'operand en el lloc del camp d'adreça

•Per especificar constants usades com operands en l'operació. Quan es treballa amb **constants** que poden ser subministrades en camp d'adreça en lloc d'accedir-les des de la memòria, estalvia, per tant, accessos a memòria.

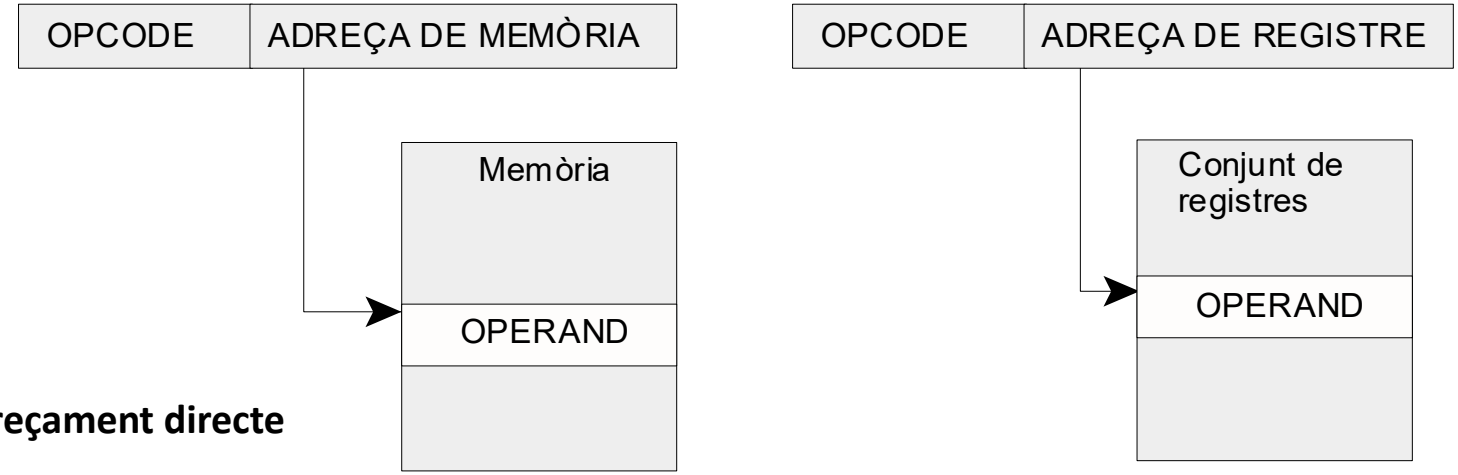
Qualsevol instrucció
Tipus I en RIPES comporta
Aquest mode d'adreçament



Adreçament IMMEDIAT

3. Mode d'adreçament directe

•El Camp Adreça especifica la posició de l'operand
•Adreçament directe a memòria o a registre del CR
•Cal tenir en compte que l'adreça de memòria sempre serà molt més gran que la del CR



Adreçament DIRECTE

Exemple mode d'adreçament directe amb RIPES:

J adreça

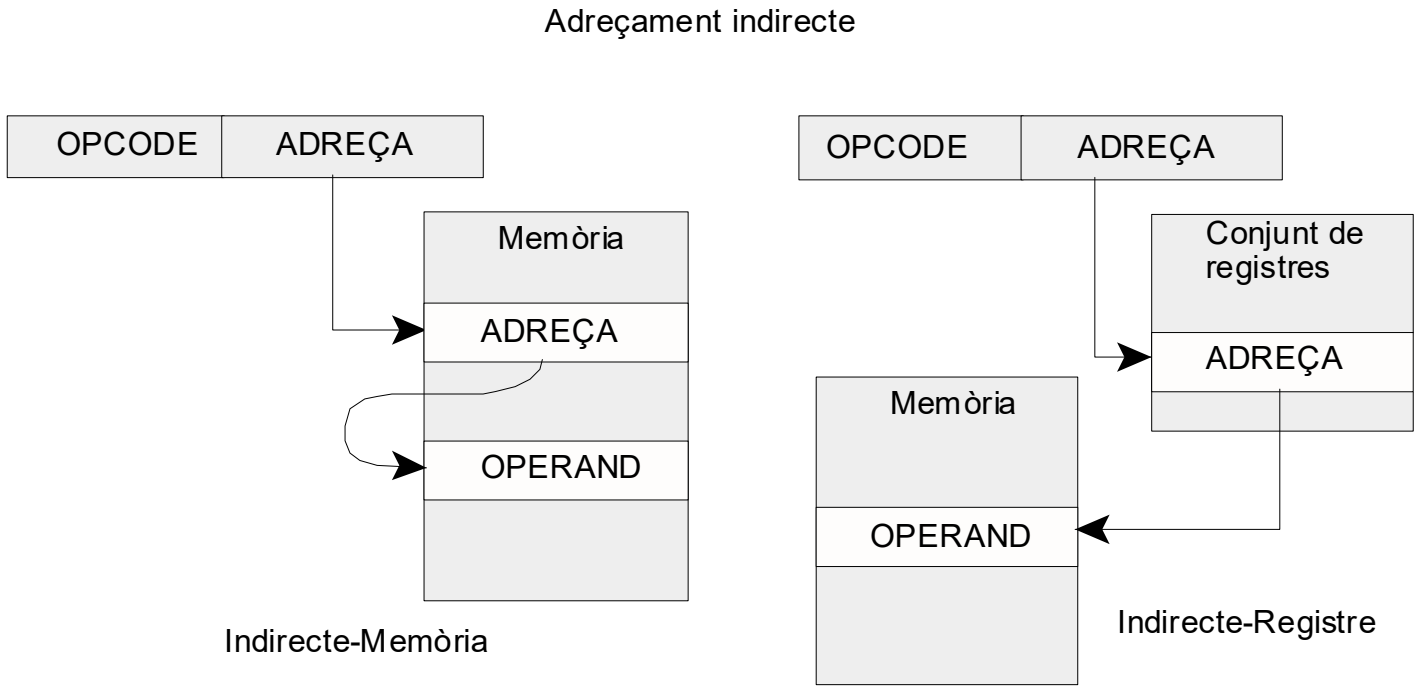
4. Mode d'adreçament indirecte

•Camp adreça especifica la ubicació de l'adreça de l'operand

•El processador ha d'accedir a memòria el doble de cops que en l'adreçament directe, 1 per llegir l'adreça i 1 per llegir l'operand o guardar el resultat

•En el mode **indirecte-registre** el camp d'adreça conté la direcció del registre que conté l'adreça de l'operand; el programador s'ha d'assegurar que l'adreça està en el registre adequat, abans d'accedir-hi

4. Mode d'adreçament indirecte



Adreçament indirecte

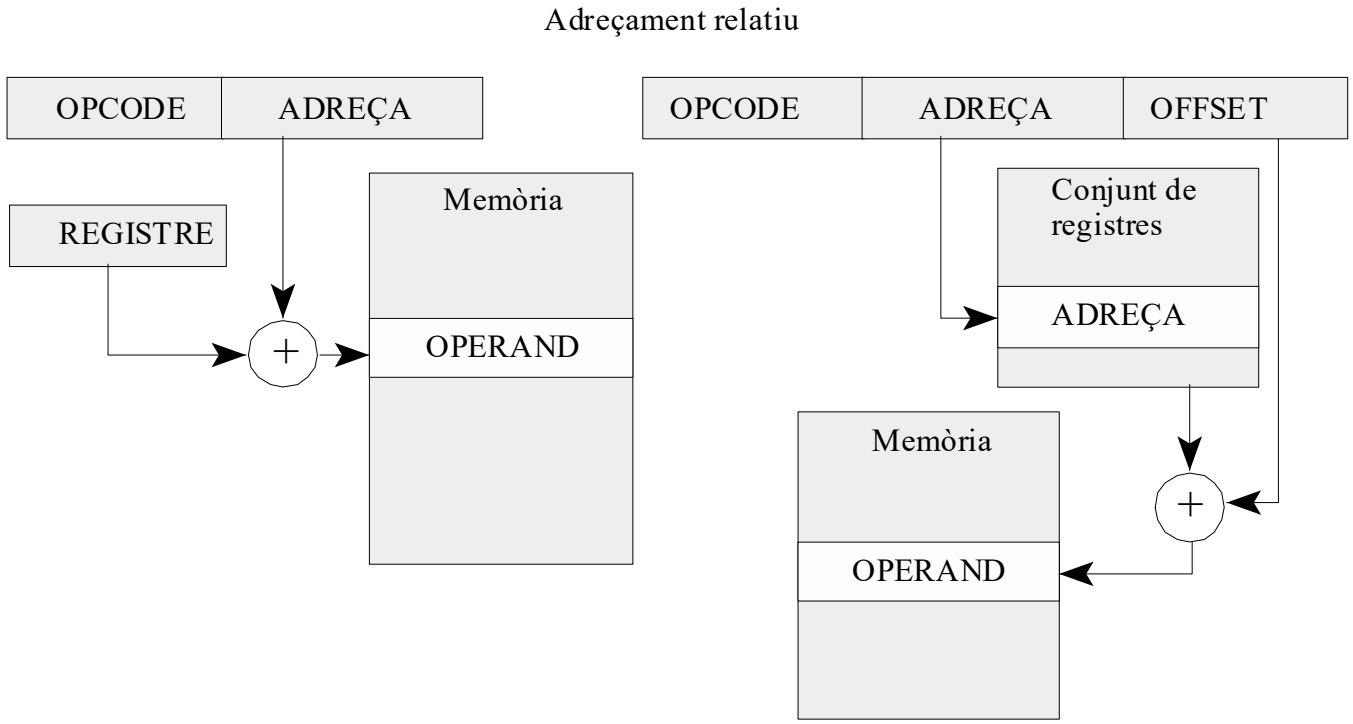
5. Mode d'adreçament relatiu

•Camp adreça=*offset*, és sumat al contingut d'un registre especificat, com el (PC) o un registre del (CR).
•L'*offset* és enter (positiu o negatiu). Quan l'*offset* es suma al PC la suma és l'adreça d'una instrucció de posició propera a la instrucció apuntada pel PC.
•Per tant aquest mode és usat en instruccions de salt condicional, ja que l'adreça de salt condicional està generalment a prop de la pròpia instrucció de salt.
•També l'adreçament relatiu es pot utilitzar respecte a qualsevol registre del CR, en aquest cas implementa taules *Look-up* en què el registre conté l'origen de la taula i l'*offset* apunta a un element específic

Exemple 2

Feu un programa senzill en ASM que incorpori aquest tipus d'adreçament. Identifiqueu-lo i expliqueu-lo

5. Mode d'adreçament relatiu



Adreçament RELATIU

6. Mode d'adreçament Indexat

•L'adreçament indexat s'usa quan es necessita accedir a dades guardades en vectors (*arrays*), piles o cues.
•L'adreça especifica una adreça inicial, anomenada base, l'índex d'una dada particular s'especifica en un registre dedicat d'índex o en un del CR.
•Càlcul d'adreça efectiva: R(base) + R(índex)
•En algunes instruccions el valor del R(índex) és incrementat o decrementat automàticament per accedir al proper element del vector.
•Aquest tipus d'instrucció s'anomena instrucció autoincrement o autodecrement. També contribueix a reduir el nombre de bits necessaris en el camp adreça.

6. Mode d'adreçament Indexat

•El mode indexat és similar al relatiu, difereixen només en les posicions de la base i de l'índex o l'*offset*.

•En relatiu la base està en un registre dedicat i l'*offset* està en el camp adreça;
•En indexat la base està en el camp adreça i l'índex està en un registre.
•El mode relatiu es sol utilitzar en salts i l'indexat en accés a dades.

6. Mode d'adreçament Indexat