# Introducció als Computadors

Tema 10: Unitat de Control General http://personals.ac.upc.edu/enricm/Docencia/IC/IC10b.pdf

Enric Morancho (enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors Facultat d'Informàtica de Barcelona Universitat Politècnica de Catalunya



2020-21, 1<sup>er</sup> quad.

Presentació publicada sota Ilicència Creative Commons 4.0 @ (1)

# Analogia: pianista vs. instruccions SISA



- Podem equiparar un piano a una UP amb 88 senyals de control
  - Les pianoles utilitzen un rodet amb paraules de control de 88 bits
    - El chulapo fa de *clock* i el rodet és la ROM amb el *programa*



1]

- Un pianista només té 10 dits
  - No pot prémer alhora algunes combinacions de tecles com fa la pianola
- La UPG té una paraula de control de 43 bits
  - $\bullet$  La UCG convertirà instruccions SISA de 16 bits en paraules de control
  - Moltes paraules de control no es podran generar amb 1 instrucció SISA

#### Índex



#### Introducció

- Ampliació de la ALU per a implementar MOVHI
- Implementació de la Lògica de Control
- Exercicis
- Conclusions

#### Objectiu

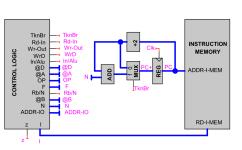


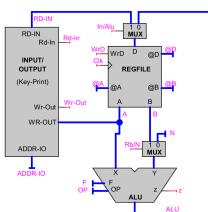
- Acabarem el disseny de la UCG
  - Ampliarem la ALU per poder implementar instrucció MOVHI
  - Dissenyarem la Lògica de control
    - Generarà la paraula de control de la UPG a partir de la codificació SISA
    - Indicarà com actualitzar el registre d'estat de la UCG, el registre PC
    - Disseny ad-hoc

# Esquema lògic UCG+UPG+IO









#### Índex



- Introducció
- Ampliació de la ALU per a implementar MOVHI
- Implementació de la Lògica de Control
- Exercicis
- Conclusions

#### Ampliació de la ALU



- Del repertori d'instruccions SISA, n'hi ha dues que la UPG actual no implementa to posan treballar amb
  - MOVI Rd, N8 A Koron

• Semàntica: Rd ← SE(N8)

• Semàntica: Rd ← SE(N8)

• Requereix fer una extensió de signe de 8 bits a bits abans d'escriure la

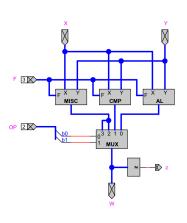
- dada al REGEILE
  - La implementarem a la lògica de control
- MOVHI Rd. N8
  - Semàntica: Rd<15...8> ← N8
  - Els bits baixos de Rd (del 0 al 7) mantenen el seu valor
  - Requereix llegir un registre font i modificar els bits alts (del 8 al 15) abans d'escriure la dada al REGEILE
  - Ampliarem l'ALU perquè la implementi

## Esquema ALU i taula de funcionalitats



- Associem a la nova funcionalitat una combinació OP/F sense ús
  - MOVHI ←⇒ OP=10 i F=010

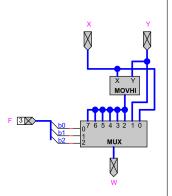
	F			0	P										
b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	11	1 0	0 1	00									
0	0	0		X	CMPLT (X, Y) CMPLE (X, Y)	AND (X, Y) OR (X, Y)									
0	1	0		MOVHI(X, Y)	(X, 1)	XOR(X, Y)									
0	1	1			CMPEQ (X, Y)	NOT (X)									
1	0	0			CMPLTU (X, Y)	ADD (X, Y)									
1	0	1			CMPLEU (X, Y)	SUB (X, Y)									
1	1	0				SHA(X, Y)									
1	1	1		<i>A</i> -		SHL(X, Y)									
	afriegum														



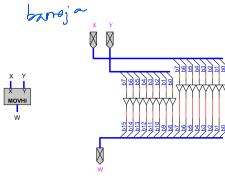
#### Blocs MISC i MOVHI



#### Bloc MISC



#### Bloc MOVHI



$$W_{7..0} \leftarrow X_{7..0}, \ W_{15..8} \leftarrow Y_{7..0}$$

#### Índex

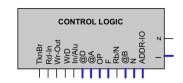


- Introducció
- Ampliació de la ALU per a implementar MOVHI
- Implementació de la Lògica de Control
- Exercicis
- Conclusions

#### Lògica de control



- La Lògica de control és un CLC
  - Entrades:
    - Instrucció SISA (16 bits)
    - Bit de condició z (1 bit)
  - Sortides:
    - Paraula de control de la UPG (43 bits)
    - Senyal TknBr (1 bit)
  - Síntesi de la Lògica de control
    - Generar la ROM és inviable
      - Tindria 2<sup>17</sup> files × 44 bits/fila, major que la I-MEM!!
    - Estudiarem com generar la paraula de control camp a camp i TknBr a partir de la codificació de les instruccions
      - Farem disseny ad-hoc
      - Camps a generar: @A, @B, @D, ADDR\_IO, N, F, OP, WrD, Rd-In, Wr-Out, Rb/N, In/Alu, TknBr



# Determinació camps ADDR IO, @A, i @B



 Camps directes: la codificació SISA els ubica a posicions fixes

ADDR\_IO: bits 7 al 0

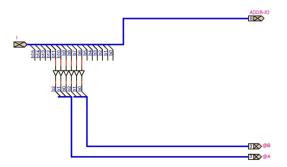
• @A: bits 11 al 9

• @B: bits 8 al 6

- Per a algunes instruccions aquests camps no seran significatius
  - Garantirem que no es provoquin efectes laterals indesitjats
  - Per exemple:
    - A les instruccions amb format 1-R, o la funció que calcula la ALU no utilitzarà el registre @B (BZ,...) o el resultat calculat per la ALU no s'emmagatzemarà enlloc (IN)
    - Si es llegeix innecesàriament un port d'entrada/sortida, el valor no s'emmagatzemarà al REGFILE
    - ...

# Determinació camps ADDR\_IO, @A, i @B





## Determinació camp @D



# Dés molt variable

• 3 opcions:

bits 5 al 3: AL, CMP

bits 8 al 6: ADDI

bits 11 al 9: MOV\*, IN

xxx la resta de casos

Formato 3-R

Formato 2-R

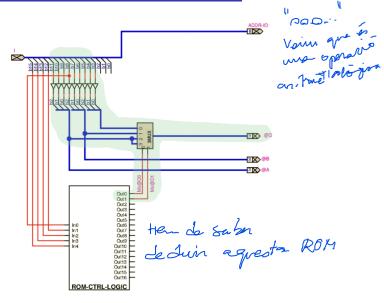
Formato 1-R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
С	С	С	С	а	а	а	b	b	b	d	d	d	f	f	f
С	С	С	С	а	а	а	d	d	d	n	n	n	n	n	n
С	С	С	С	a d	a d	a d	е	n	n	n	n	n	n	n	n

- Enviarem les possibles opcions a un multiplexor
- Tindrem una ROM, la ROM-CTRL-LOGIC, que generarà el senyal de selecció en funció del codi d'operació
  - Cinc bits d'entrada:
    - els quatre bits del codi d'operació
    - el bit d'extensió del codi d'operació
  - La ROM-CTRL-LOGIC tindrà 32 files
    - Veurem quantes columnes ens faran falta

# Determinació camp @D





## Determinació camp N



• 3 processaments possibles:

SE(bits 7 al 0): MOV\*

2.SE(bits 7 al 0): B\*

• XXXX la resta de casos

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Formato 3-R	С	С	С	С	а	a	a	b	b	b	d	d	d	f	f	
Formato 2-R	С	С	С	С	а	а	а	d	d	d	n	n	n	n	n	
Formato 1-R	С	С	С	С	a d	a d	a d	е	n	n	n	n	n	n	n	

• En tots els casos, es genera una dada de 16 bits

• Apliquem extensió de signe fins a tenir 16 bits

• En el cas de BZ i BNZ, a més cal multiplicar per 2

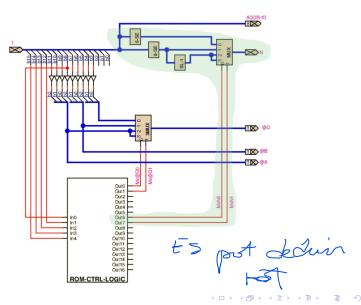
• Perquè la I-MEM estarà direccionada a byte

 Afegirem un multiplexor per triar entre les tres opcions i hardware per a generar els resultats

La ROM-CTRL-LOGIC generarà els senyals de selecció

#### Determinació camp N





#### Determinació camp F



Diverses opcions:

bits 2 al 0: AL, CMP

100: ADDT

• 000: B\*

001 · MOVT

010: MOVHI

xxx la resta de casos

Formato 3-R

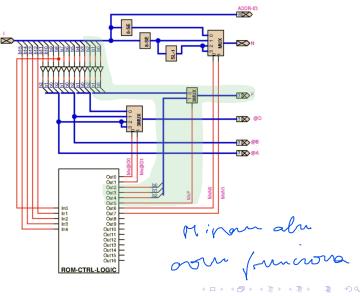
ccclaaalddd Innnnnn Formato 2-R Formato 1-R

15 14 13 12 11 10 9 8 7 6 5 4

- En alguns casos ho extreurem de la instrucció, en d'altres de la ROM
- Afegirem un multiplexor per triar entre les dues possibilitats
  - La ROM-CTRL-LOGIC generarà el senyal de selecció del multipexor i el valor de F per a les instruccions que no siguin AL ni CMP

#### Determinació camp F





# Determinació camps OP, In/Alu, Rb/N



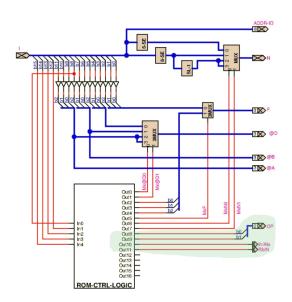
Només depenen del codi d'operació i de l'extensió del codi d'operació:

In 1Ahr només val. 1 Si ve de l'externor

- OP:
  - 00: AL, ADDI
  - 01: CMP
  - 10: MOV\*
  - x a la resta de casos
- In/Alu:
  - 1: IN
  - 0: AL, CMP, ADDI, MOV\*
  - x a la resta de casos
- Rb/N:
  - 1: AL, CMP
  - O: ADDI, MOV\*
  - x a la resta de casos
- La ROM-CTRL-LOGIC generarà tots aquests senyals

# Determinació camps OP, In/Alu, Rb/N





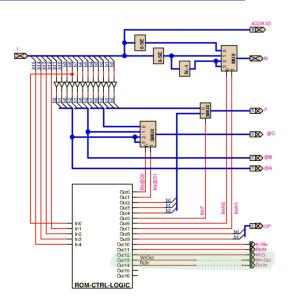
#### Determinació camps WrD, Wr-Out, Rd-In



- Camps que controlen la modificació de l'estat del computador
- Només depenen del codi d'operació i de l'extensió del codi d'operació
  - WrD:
    - 1: AL, CMP, ADDI, MOV\*, IN
    - 0 a la resta de casos
  - Wr-Out:
    - 1: OUT
    - 0 a la resta de casos
  - Rd-In:
    - 1 · TN
    - 0 a la resta de casos
- La ROM-CTRL-LOGIC generarà tots aquests senyals
- Com controlen la modificació de l'estat del computador, mai valdran x
- A l'esquema de la Lògica de Control, a Wr-Out i Rd-In apareixen uns punts suspensius
  - Problemes de temporalitat... en parlarem (i solucionarem) al tema 12

## Determinació camps WrD, Wr-Out, Rd-In





## Determinació camp TknBr

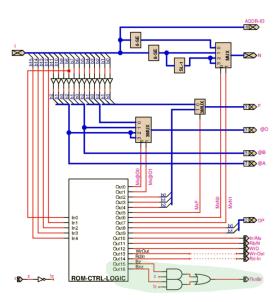


Hu de Scresa TKnBr pels solts.

- Control del seqüenciament
- ullet Depèn del codi d'operació, de l'extensió de codi d'operació i del bit z
  - TknBr:
    - z: BZ
    - !z: BNZ
    - 0 a la resta de casos
- La ROM-CTRL-LOGIC indicarà si la instrucció és BZ i BNZ
  - Un petit CLC calcularà TknBr

## Determinació camp TknBr





# Contingut ROM-CTRL-LOGIC



• Cal saber deduir el seu contingut a partir dels esquemes lògics

• Poseu x allà on sigui possible

U	u	$\sim$		J	<u>ج</u> ر	41	$\mathbf{r}$	-	,J	$\mathbf{v}$	_											_	/	, ,
	ln4	ln3	ln2	In T	0 U	Out16	Out15	Out14	Out13	Out12	Out11	Out10	Out9	Out8	Out7	Out6	Out5	Out4	Out3	Out2	Out1	2000		ollo
	I<15>	I<14>	I<13>	I<12>	\ 8>I	Bnz	Bz	RdIn	WrOut	WrD	Rb/N	In/Alu	OP1	OP0	M×N1	M×N0	MxF	F2	Œ	F0	Mx@D1	O BOYIN		
				0																			AL	
	0	0	0	1	х																		CMP	
	0	0	1	0	х																		ADDI	
	1	0	0	0	0																	1	BZ	
	1	0	0	0	1																		BNZ	
	1	0	0	1	0																		MOVI	
	1	0	0	1	1																		MOVHI	
	1	0	1	0	0																		IN	
	1	0	1	0	1																		OUT	
	1	0	1	1	0																		NOP	

# Contingut ROM-CTRL-LOGIC



• Cal saber deduir el seu contingut a partir dels esquemes lògics ·

Si terim

Jutis

mram

pux-pan

puletan

in straanons • Poseu x allà on sigui possible exercici on per to port B7. BNZ MOVI 7 MOVHI IN TUO NOP

#### Índex



- Introducció
- Ampliació de la ALU per a implementar MOVHI
- Implementació de la Lògica de Control
- Exercicis
- Conclusions

#### Exercicis típics



- Deduir el contingut de la ROM-CTRL-LOGIC
  - Es pot demanar alguna(es) fila(s), columna(es) o interseccions
  - Practiqueu!!
  - Poseu x sempre que sigui possible

#### Exercicis a entregar a Atenea



- Enunciat disponible a Atenea
  - https://atenea.upc.edu/pluginfile.php/3603443/mod\_ assign/introattachment/0/Tema%2010%20-%20Exercicis%20en% 20paper.pdf?forcedownload=1
- Entrega a Atenea fins el dimecres 25/11
  - Format PDF
  - Per fer els grafs d'estats us pot resultat útil l'editor on-line https://www.cs.unc.edu/~otternes/comp455/fsm\_designer/

#### Índex



- Introducció
- Ampliació de la ALU per a implementar MOVHI
- Implementació de la Lògica de Control
- Exercicis
- Conclusions

#### Conclusions



- Hem definit llenguatge màquina SISA
  - Codifica les instruccions amb 16 bits
    - Codificació compacta, sense x's
  - Inicialment, el repertori té 20 instruccions
- Hem creat la Lògica de control
  - A partir de la codificació en 16 bits, genera la paraula de control
  - És un CLC dissenyat ad-hoc amb una ROM-CTRL-LOGIC i hardware
    - La ROM-CTRL-LOGIC té 32 files  $\times$  17 columes = 544 bits
    - ullet La síntesi completa amb una ROM hagués requerit pprox 5,5 Megabits
- Al tema següent incorporarem al computador una memòria de dades
  - Afegirem 4 instruccions al repertori SISA
  - Caldrà adaptar la Lògica de Control
- No oblideu fer el qüestionari ET10b i els exercicis en paper (slide 30)

#### Referències I



Llevat que s'indiqui el contrari, les figures, esquemes, cronogrames i altre material gràfic o bé han estat extrets de la documentació de l'assignatura elaborada per Juanjo Navarro i Toni Juan, o corresponen a enunciats de problemes i exàmens de l'assignatura, o bé són d'elaboració pròpia.

[1] [Online]. Available: https://instrumundo.blogspot.com/2013/06/organillo.html.

# Introducció als Computadors

Tema 10: Unitat de Control General http://personals.ac.upc.edu/enricm/Docencia/IC/IC10b.pdf

> Enric Morancho (enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors Facultat d'Informàtica de Barcelona Universitat Politècnica de Catalunva



2020-21, 1<sup>er</sup> quad.

Presentació publicada sota Ilicència Creative Commons 4.0 @(1)(\$)(=)

