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

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.

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]

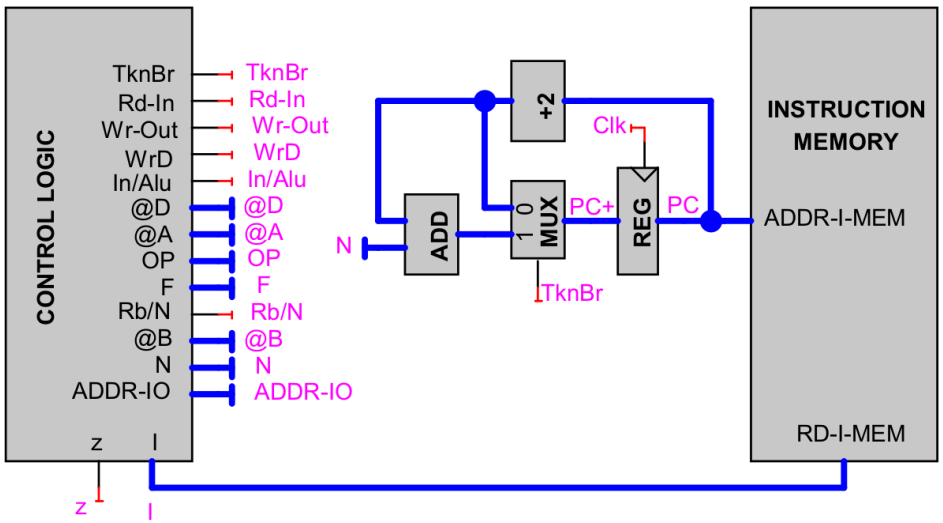
  - Un pianista només té 10 dits
    - No pot prémer alhora algunes combinacions de tecles com fa la pianola
- La UPG té una paraula de control de 43 bits
  - La UCG convertirà instruccions SISA de 16 bits en paraules de control
  - Moltes paraules de control no es podran generar amb 1 instrucció SISA

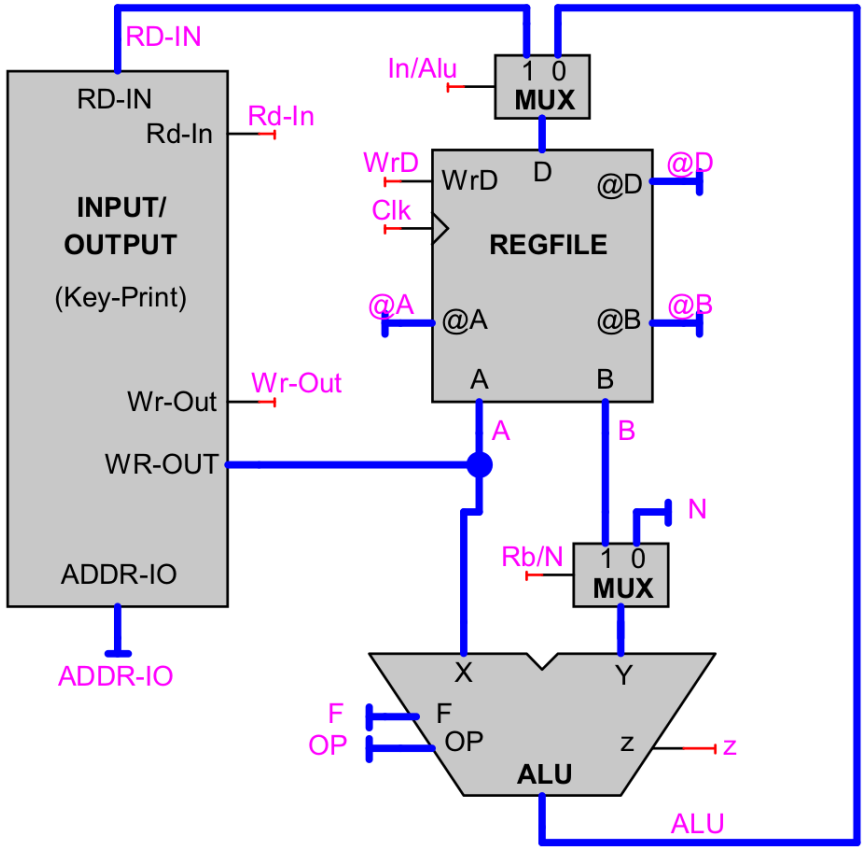
Índex

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

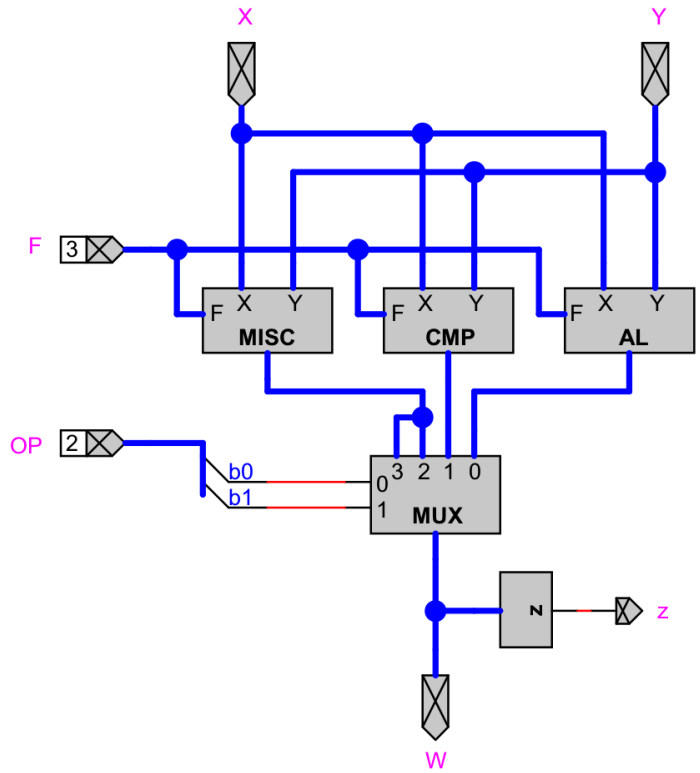
Ampliació de la ALU

- Del repertori d'instruccions SISA, n'hi ha dues que la UPG actual no implementa
  - `MOVI Rd, N8`
    - Semàntica: `Rd` $\leftarrow$ `SE(N8)`
    - Requereix fer una extensió de signe de 8 bits a bits abans d'escriure la dada al REGFILE
    - La implementarem a la lògica de control
  - `MOVHI Rd, N8`
    - Semàntica: `Rd<15..8>` $\leftarrow$ `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 REGFILE
    - Ampliarem l'ALU perquè la implementi

Esquema ALU i taula de funcionalitats

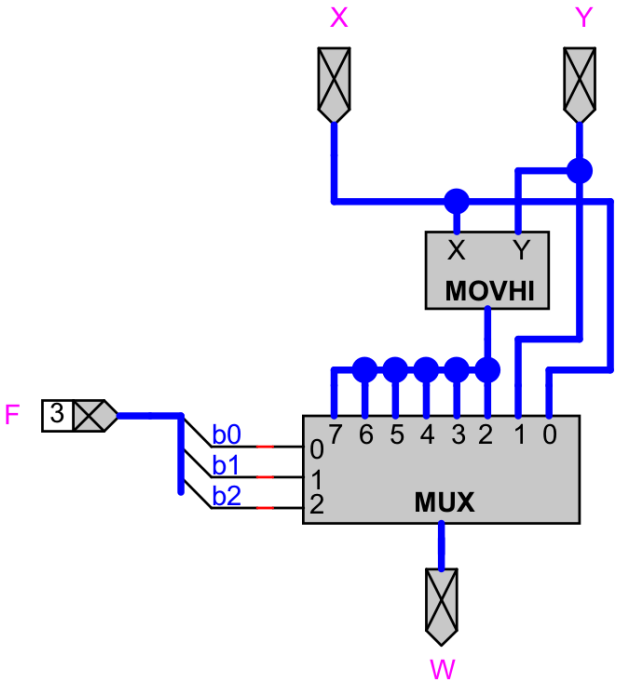
- Associem a la nova funcionalitat una combinació OP/F sense ús
  - `MOVHI` $\Longleftrightarrow$ OP=10 i F=010

F			OP			
b_2	b_1	b_0	11	10	01	00
0	0	0	---	X	CMPLT (X, Y)	AND (X, Y)
0	0	1	---	Y	CMPLE (X, Y)	OR (X, Y)
0	1	0	---	MOVHI(X, Y)	---	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	---	---	---	SHL(X, Y)

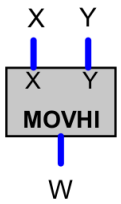


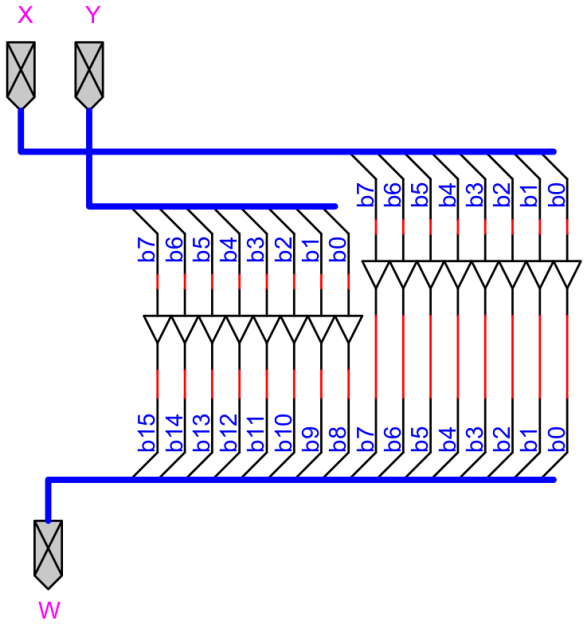
Blocs MISC i MOVHI

Bloc MISC



Bloc MOVHI



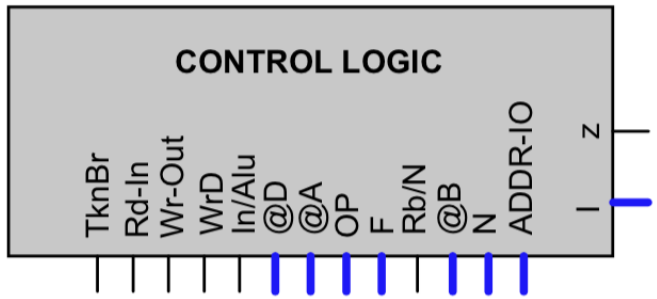


$$W_{7..0} \leftarrow X_{7..0}\,, \quad W_{15..8} \leftarrow Y_{7..0}$$

Índex

Lògica de control

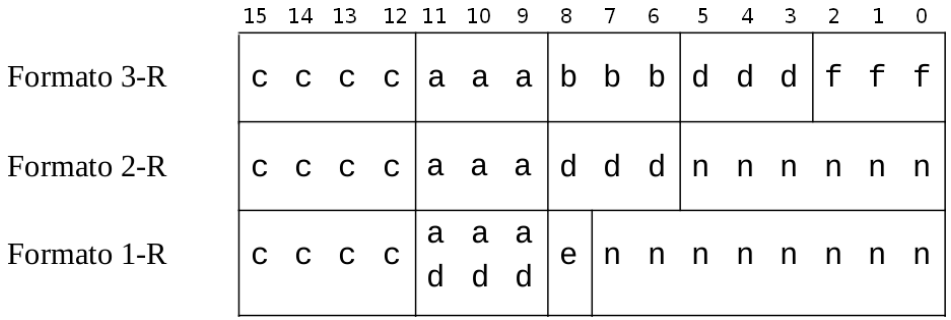
- La Lògica de control és un CLC
  - Entrades:
    - Instrucció SISA (16 bits)
    - Bit de condició z (1 bit)
  - Sortides:
    - Paraula de control de la UPG (43 bits)
    - Senyal TknBr (1 bit)



- Síntesi de la Lògica de control
  - Generar la ROM és inviable
    - Tindria 2^{17} files $\times$ 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

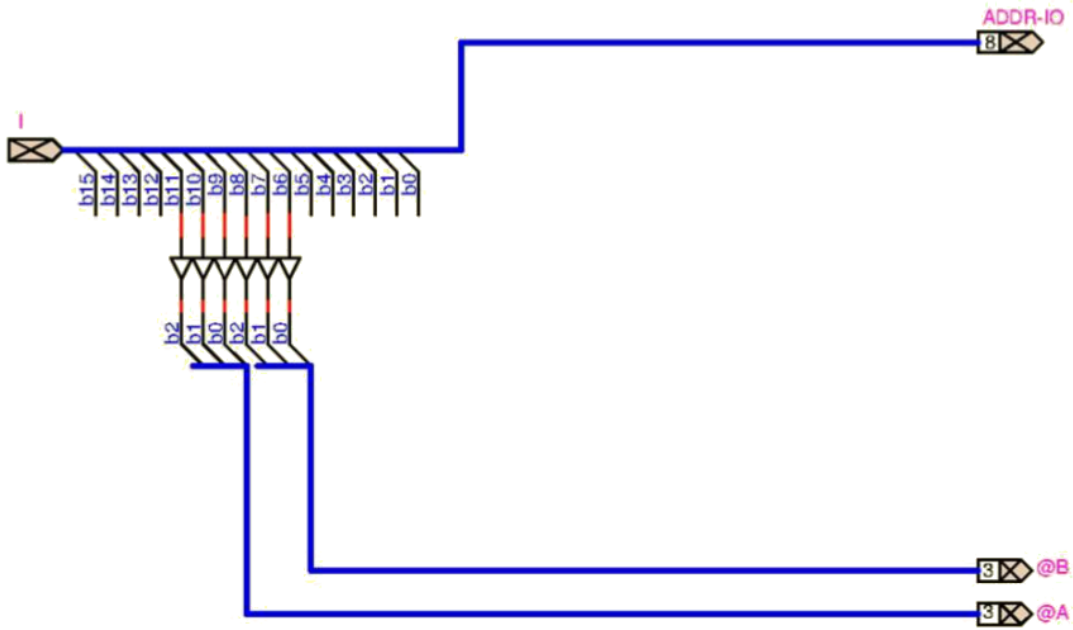
- Camps directes: la codificació SISA els ubica a posicions fixes
  - ADDR_IO: bits 7 al 0
  - @A: bits 11 al 9
  - @B: bits 8 al 6



- 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
    - ...

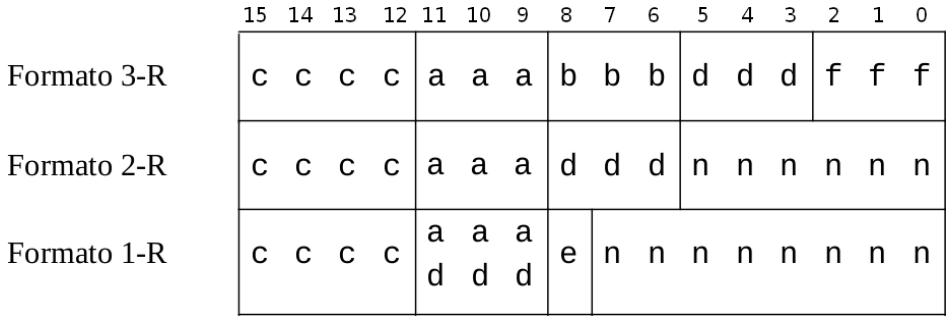
Determinació camps ADDR_IO, @A, i @B

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona



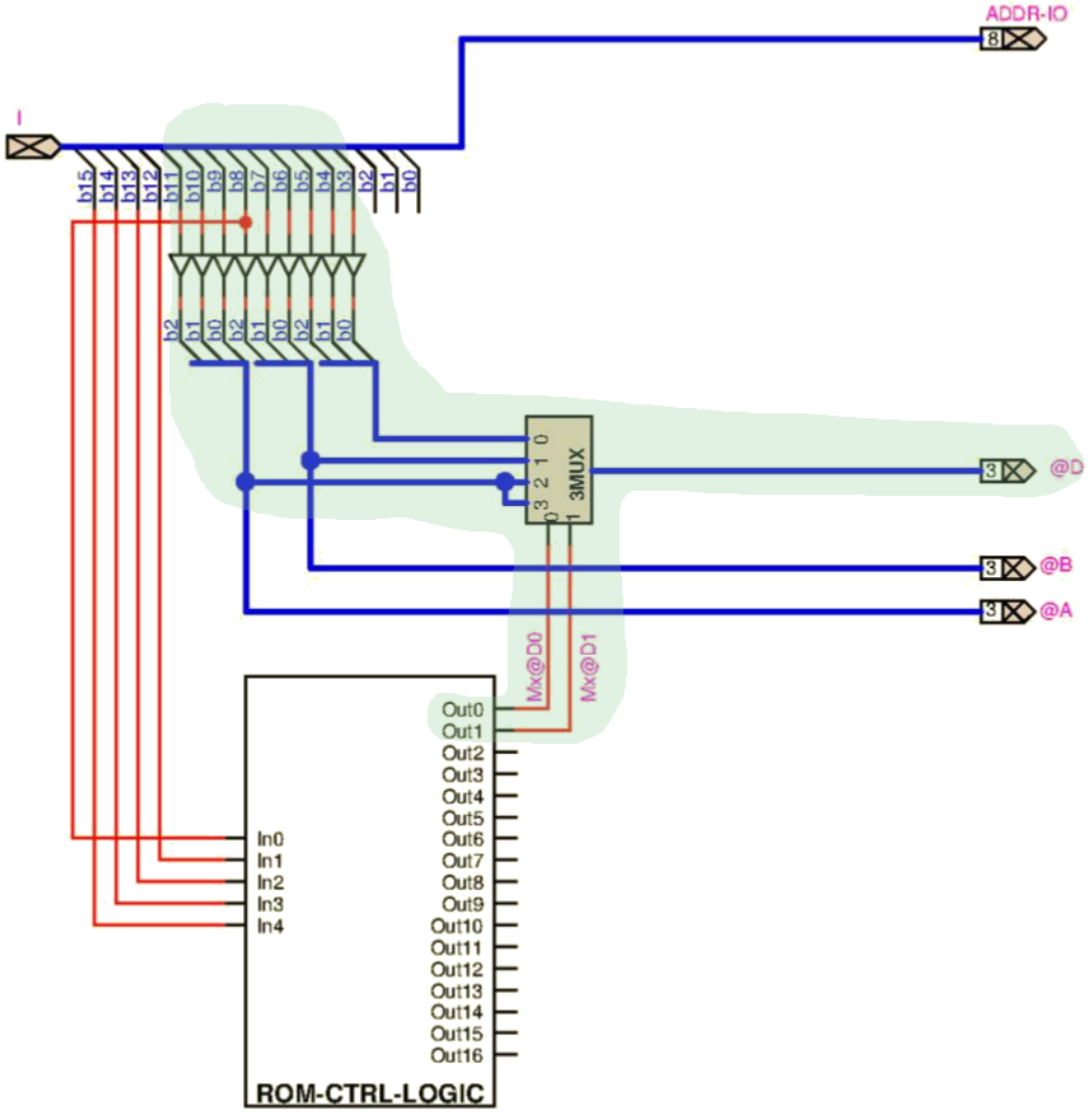
Determinació camp @D

- 3 opcions:
  - bits 5 al 3: `AL, CMP`
  - bits 8 al 6: `ADDI`
  - bits 11 al 9: `MOV*, IN`
  - `xxx` la resta de casos



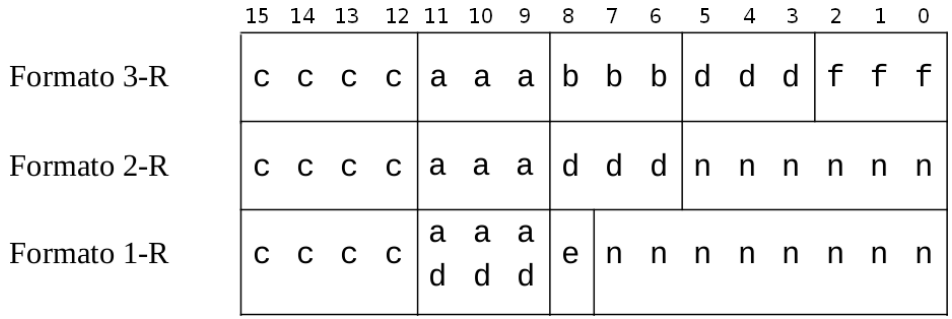
- Enviarem les possibles opcions a un multiplexor
- Tindrem una ROM, la ROM-CTRL-LOGIC, que generarà el senyal de selecció en funció del codi d'operació
  - Cinc bits d'entrada:
    - els quatre bits del codi d'operació
    - el bit d'extensió del codi d'operació
  - La ROM-CTRL-LOGIC tindrà 32 files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

Determinació camp @D



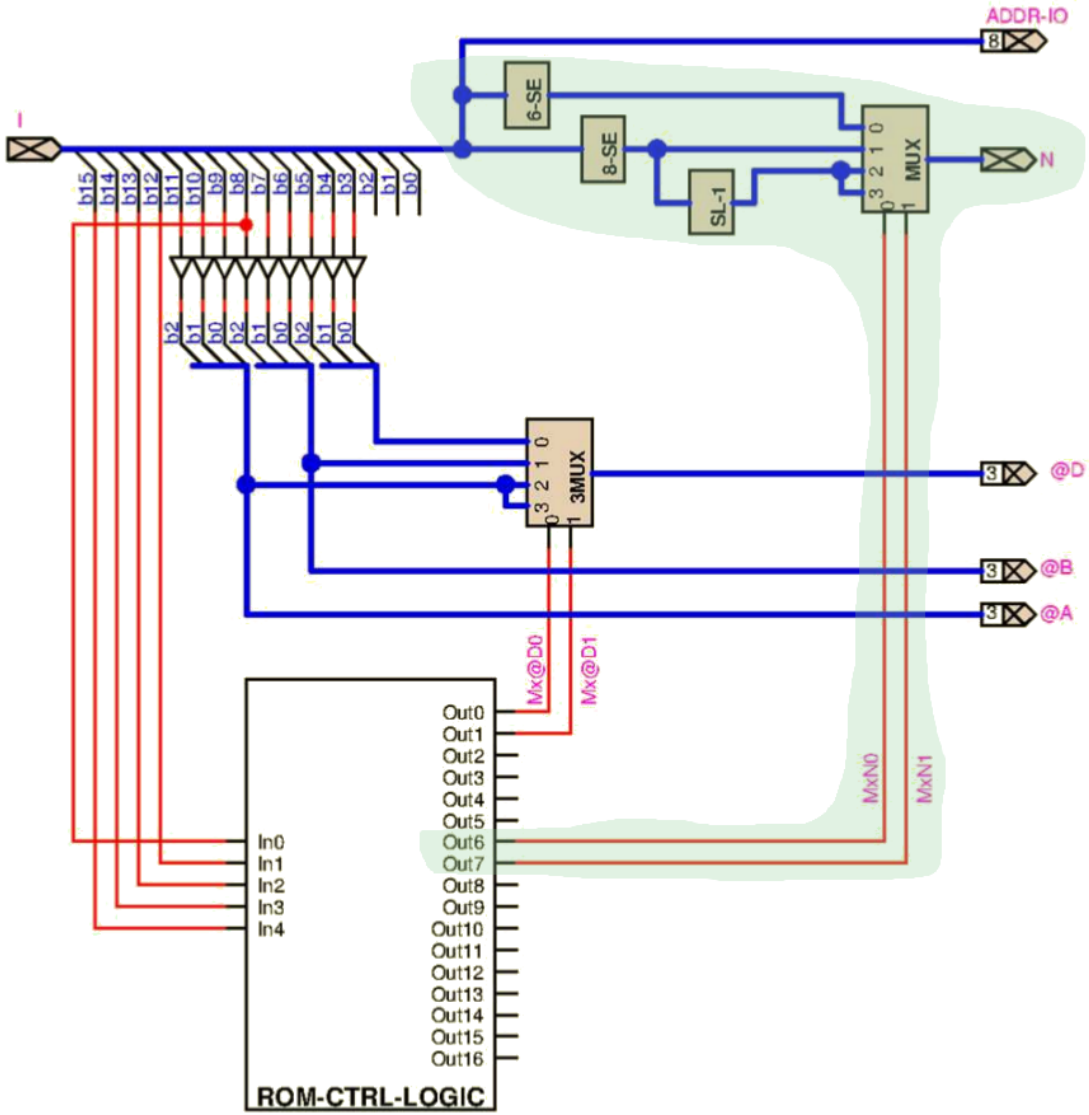
Determinació camp N

- 3 processaments possibles:
  - SE(bits 5 al 0): `ADDI`
  - SE(bits 7 al 0): `MOV*`
  - 2·SE(bits 7 al 0): `B*`
  - `XXXX` la resta de casos



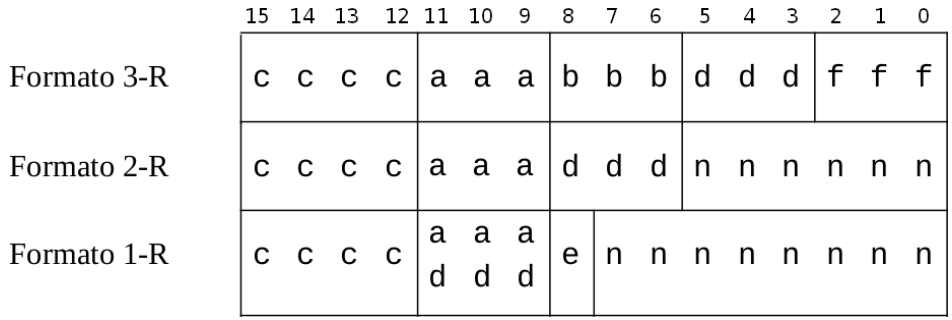
- En tots els casos, es genera una dada de 16 bits
  - Apliquem extensió de signe fins a tenir 16 bits
- En el cas de `BZ` i `BNZ`, a més cal multiplicar per 2
  - Perquè la I-MEM estarà direccionada a byte
- Afegirem un multiplexor per triar entre les tres opcions i *hardware* per a generar els resultats
  - La ROM-CTRL-LOGIC generarà els senyals de selecció

Determinació camp N



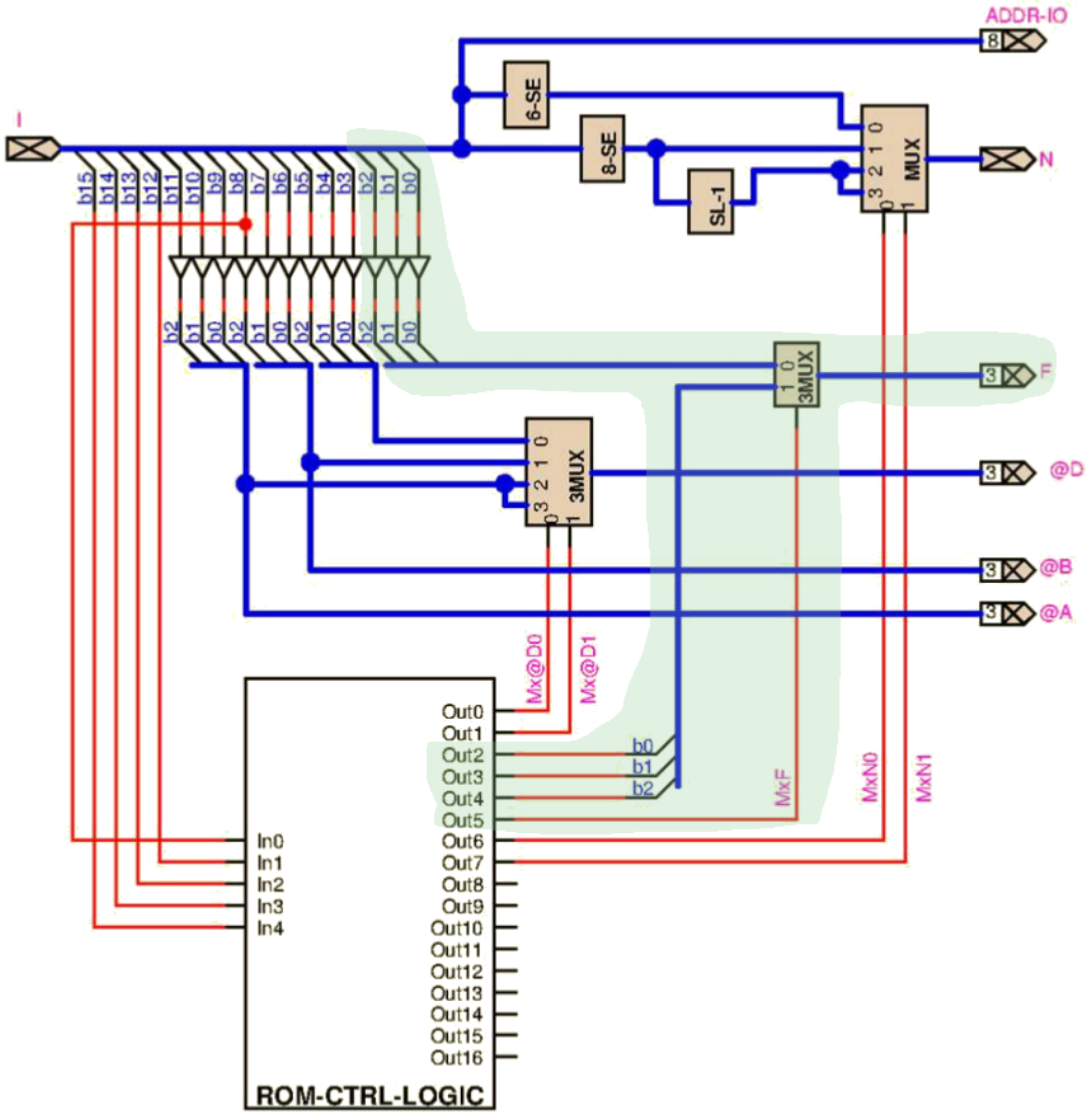
Determinació camp F

- Diverses opcions:
  - bits 2 al 0: `AL, CMP`
  - 100: `ADDI`
  - 000: `B*`
  - 001: `MOVI`
  - 010: `MOVHI`
  - `xxx` la resta de casos



- 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

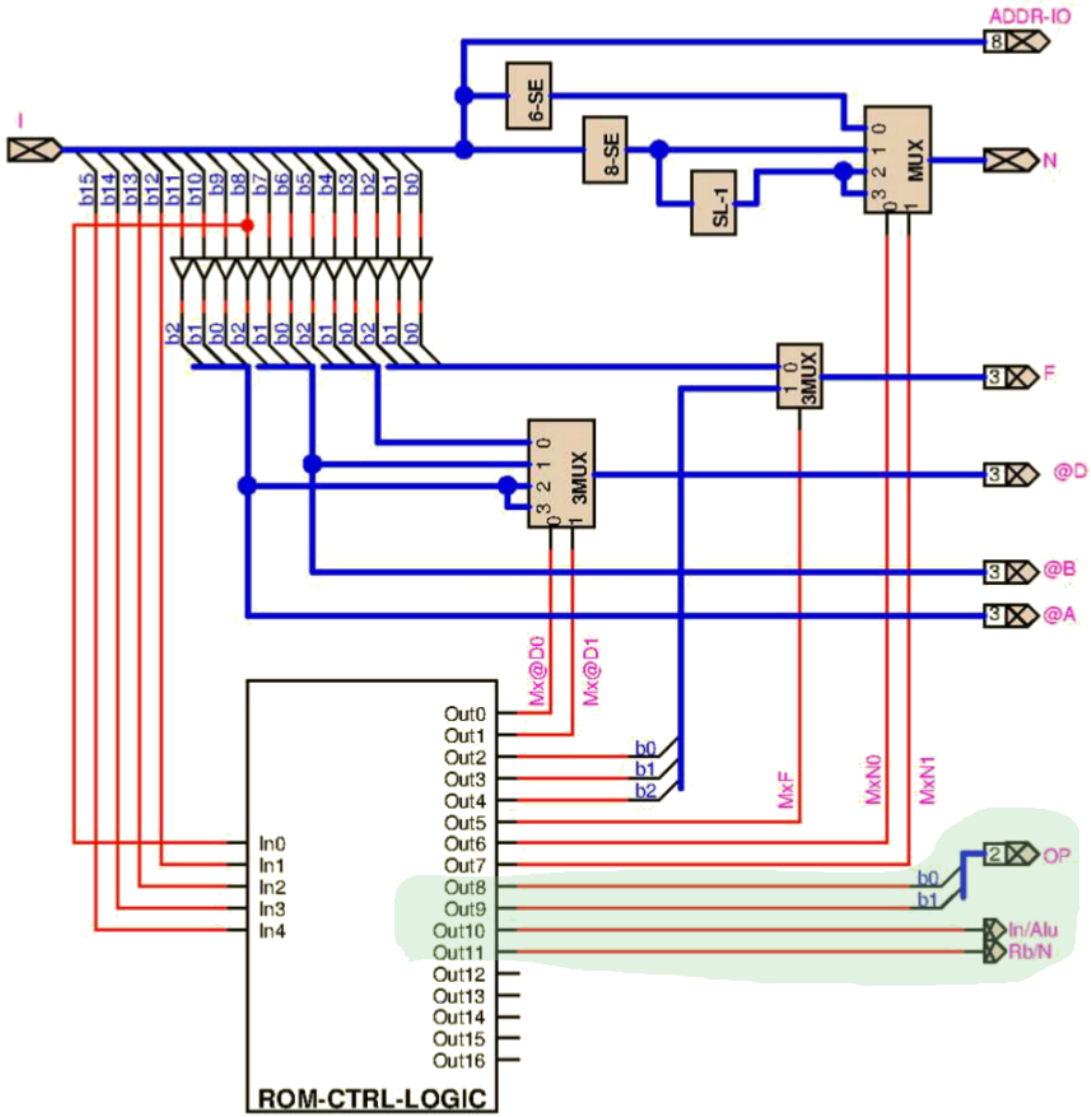
Determinació camp F



Determinació camps OP, In/Alu, Rb/N

- Només depenen del codi d'operació i de l'extensió del codi d'operació:
  - 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`
    - 0: `ADDI, MOV*`
    - `x` a la resta de casos
- La ROM-CTRL-LOGIC generarà tots aquests senyals

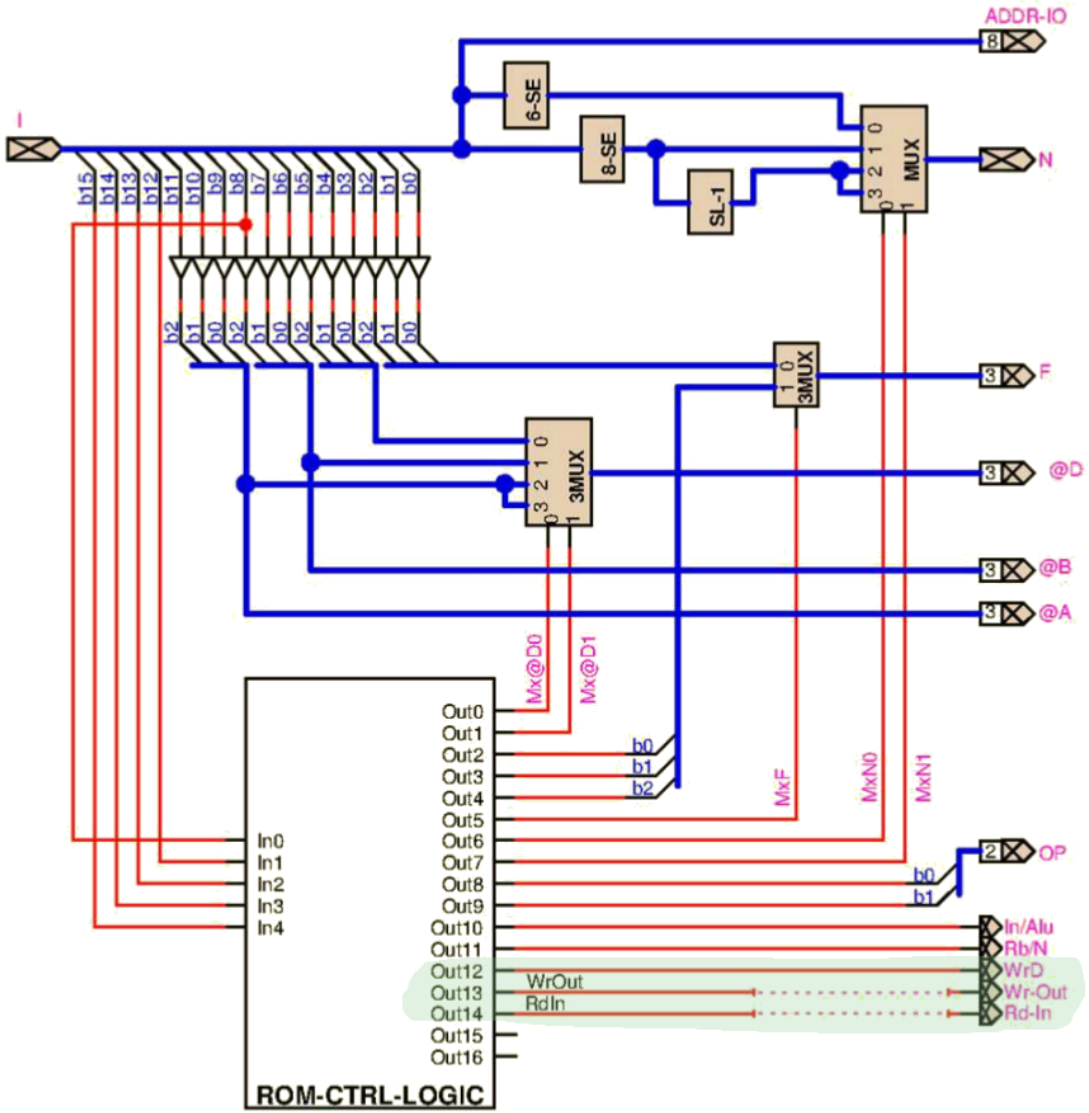
Determinació camps OP, In/Alu, Rb/N



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: `IN`
    - **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

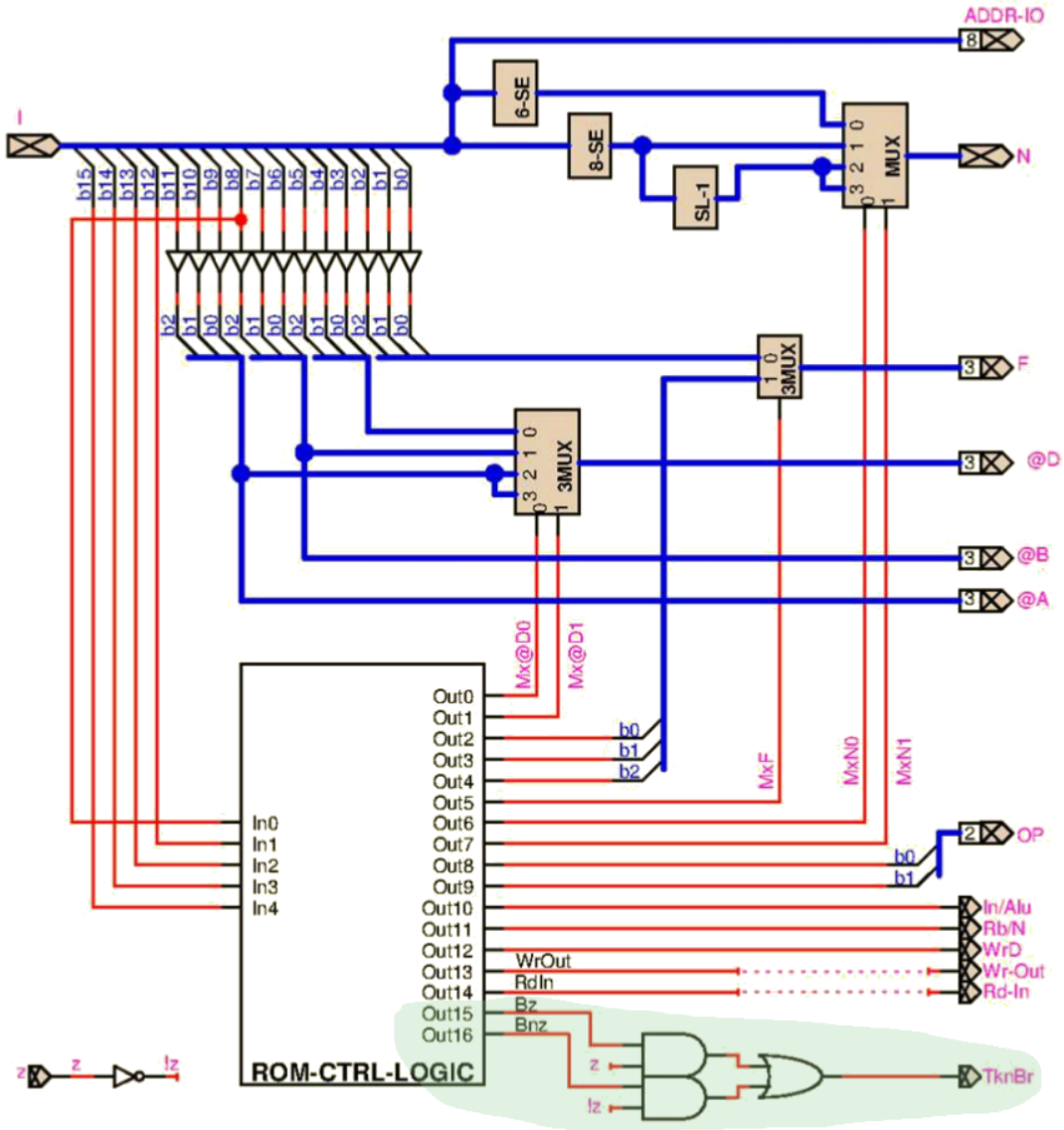
Determinació camps WrD, Wr-Out, Rd-In



Determinació camp TknBr

- Control del seqüenciament
- 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

In4 I<15>	In3 I<14>	In2 I<13>	In1 I<12>	In0 I<8>	Out16 Bnz	Out15 Bz	Out14 RdIn	Out13 WrOut	Out12 WrD	Out11 Rb/N	Out10 In/Alu	Out9 OP1	Out8 OP0	Out7 MxN1	Out6 MxN0	Out5 MxF	Out4 F2	Out3 F1	Out2 F0	Out1 Mx@D1	Out0 Mx@D0	
0	0	0	0	x																		AL
0	0	0	1	x																		CMP
0	0	1	0	x																		ADDI
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	...
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	...
1	0	0	0	0																		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
- Poseu *x* allà on sigui possible

In4	In3	In2	In1	In0	Out16	Out15	Out14	Out13	Out12	Out11	Out10	Out9	Out8	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0	
I<15>	I<14>	I<13>	I<12>	I<8>	Bnz	Bz	RdIn	WrOut	WrD	Rb/N	In/Alu	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0	0	0	0	x	0	0	0	0	1	1	0	0	0	x	x	0	x	x	x	0	0	AL
0	0	0	1	x	0	0	0	0	1	1	0	0	1	x	x	0	x	x	x	0	0	CMP
0	0	1	0	x	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	ADDI
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	...
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	...
1	0	0	0	0	0	1	0	0	0	x	x	1	0	1	0	1	0	0	0	x	x	BZ
1	0	0	0	1	1	0	0	0	0	x	x	1	0	1	0	1	0	0	0	x	x	BNZ
1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	1	1	0	MOVI
1	0	0	1	1	0	0	0	0	1	0	0	1	0	0	1	1	0	1	0	1	0	MOVHI
1	0	1	0	0	0	0	1	0	1	x	1	x	x	x	x	x	x	x	x	1	0	IN
1	0	1	0	1	0	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	OUT
1	0	1	1	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	NOP
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	...

Índex

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

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
    - La síntesi completa amb una ROM hagués requerit $\approx$ 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 Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.