Introducció als Computadors

Tema 13: Computador SISC von Neumann http://personals.ac.upc.edu/enricm/Docencia/IC/IC13b.pdf

Enric Morancho (enricm@ac.upc.edu)

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



2020-21, 1^{er} quad.

Presentació publicada sota Ilicència Creative Commons 4.0 @ (1) (3)

UCG com a intèrpret del llenguatge SISA



- El contingut de les ROMs Q+ i OUT de UCG és un programa
 - Intèrpret del llenguatge màquina SISA
 - Típicament rep el nom de microcodi/microprograma del processador
 - Tipus particular de firmware
- Excepcionalment, caldrà actualitzar el microcodi del processador
 - Incorporar noves funcionalitats al llenguatge màquina
 - Exercici estrella d'aquest tema :-)
 - Solucionar/mitigar bugs al processador
 - Els processadors comercials tenen bugs: Pentium FDIV bug (1994) [1],
 Meltdown (2017) [2], Microarchitectural Data Sampling (2018) [3], ...

entempetity-3 desigling interaceds entempetity-3 desigling interaceds a nonesign interaceds interaceds undered early to revition nor, date = 2915-64-23 a nonesign interaceds interaceds undered early to revition nor, date = 2915-64-23 a nonesign interaceds interaced undered early to revition nor demonstrated interaced int



General Information Processor Name: Intel Core 17-6700 Crininal Processor Frequency 3400 0 MHz CPU ID: 000506E3 CPU Brand Name: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU Vendori GenuineIntel CPU Steeping: CPU Code Name: Skylake-S CPU Technology: CPU S-Spect SR28T, SR2L2 G CPU Thermal Design Power (TDP): CPU Power Limits (Max): Power = Unimited, Time = Unimited CPU Power Limit 1 - Long Duration: Power = 65.00 W, Time = 8.00 sec [Unlocked] CPU Power Limit 2 - Short Duration: Power = 81.25 W, Time = 2.44 ms [Unlocked] CPU Max. Junction Temperature (T)_max): 100 °C CPU Type: Production Unit CPU Platforms Socket H4 (LGA1151)

Windows update [4]



Índex

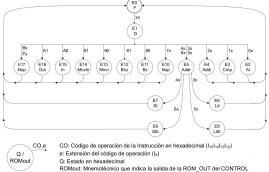


- Introducció
- Esquema lògic de la Unitat de Control
- Exercicis
- Conclusions

Introducció



- Implementarem la UC del computador Von Neumann amb un CLS
 - n=5 bits d'entrada, k=5 bits d'estat i m=24 bits de sortida
 - ROM Q+: Determinació de l'estat següent de la UC



- ROMout: N
- ROM OUT: tindrà 24 bits de sortida
 - Combinant aquests 24 bits, els 16 de la instrucció (IR) i el bit z es generaran els 51 bits de la paraula de control pròpia de cada estat

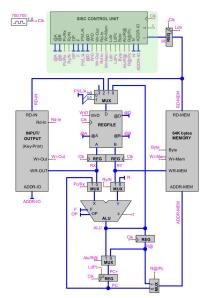
Índex



- Introducció
- Esquema lògic de la Unitat de Control
- Exercicis
- Conclusions

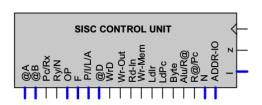
SISC CONTROL UNIT: ubicació a la UPG





Encapsulat SISC CONTROL UNIT





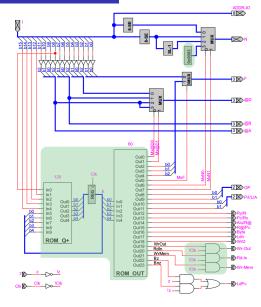
- 17 bits d'entrada
 - Instrucció SISA (I) i bit Z
- 51 bits de sortida
 - Addicions respecte a la del computador Harvard multicicle:
 - PC/Rx, Alu/R@, R@/PC: Senyals de control dels nous multiplexors
 - LdIr: Senyal de càrrega del registre IR
 - Canvis de nom:
 - Ry/N: abans era Rb/N
 - P/I/L/A: abans era -/I/L/A
 - LdPc: abans era TknBr



Esquema lògic SISC CONTROL UNIT



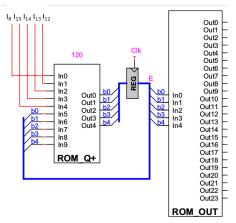
- CLS intern
 - n = 5 bits d'entrada
 - k = 5 bits d'estat
 - m = 24 bits de sortida
- Cal generar 0x0002
 - $PC \leftarrow PC + 2$
- Wr-Out, Rd-In, Wr-Mem
 - AND-2 amb !Clk
 - Com a Harvard unicicle
- Genera 51 bits a partir de:
 - 16 bits codificació SISA (I)
 - 24 bits ROM OUT
 - bit z
 - senyal Clk



CLS de la SISC CONTROL UNIT



- Parametrització: n = 5 $(I_{15}, I_{14}, I_{13}, I_{12}, I_8), k = \lceil log_2 18 \rceil = 5, m = 24$
 - Model de Moore
 - Q+ depèn de l'estat actual (k=5 bits) i de l'entrada (n=5 bits)
 - La sortida (m=24 bits) depèn de l'estat actual (k=5 bits)



ROM OUT



- Cal saber deduir el seu contingut a partir dels esquemes lògics
 - 2^k paraules \times m bits/paraula = 2^5 paraules \times 24 bits/paraula = 768 bits
 - Poseu x's on sigui possible

@ROM	Bnz B7	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	MxN1	M×N0	MxF	F2	Ε	9	Mx@D1	Mx@D0	
0													- 1											F
1													1											D
2																								Al
3																								Cmp
4													- 1											Addi
5																								Addr
6																								Ld
7																								St
8																								Ldb
9													1											Stb
10																								Jalr
11																								Bz
12													1											Bnz
13																								Movi
14													1											Movhi
15																								In
16																								Out
1731													1											Nop

ROM OUT



- Cal saber deduir el seu contingut a partir dels esquemes lògics
 - 2^k paraules \times m bits/paraula = 2^5 paraules \times 24 bits/paraula = 768 bits
 - Poseu x's on sigui possible

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	M×N1	M×N0	MxF	F2	Œ	F0	Mx@D1	Mx@D0	
0	1	1	0	0	0	0	1	0	0	1	1	0	х	х	0	0	1	1	1	1	0	0	х	х	F
1	0	0	0	0	0	0	0	х	х	х	1	0	х	х	0	0	1	0	1	1	0	0	х	х	D
2	0	0	0	0	0	1	Х	х	х	х	0	1	0	0	0	0	х	Х	0	х	х	х	0	0	Al
3	0	0	0	0	0	1	Х	х	х	х	0	1	0	0	0	1	х	х	0	х	х	х	0	0	Cmp
4	0	0	0	0	0	1	Х	х	х	Х	0	0	0	0	0	0	0	0	1	1	0	0	0	1	Addi
5	0	0	0	0	0	0	0	х	х	х	0	0	х	Х	0	0	0	0	1	1	0	0	х	х	Addr
6	0	0	0	0	0	1	X	0	1	х	х	х	0	1	х	Х	х	Х	х	х	х	х	0	1	Ld
7	0	0	1	0	0	0	Х	0	1	Х	Х	X	x	X	х	Х	х	Х	х	х	X	х	х	х	St
8	0	0	0	0	0	1	Х	1	1	Х	X	Х	0	1	х	Х	х	Х	х	x	Х	х	0	1	Ldb
9	0	0	1	0	0	0	Х	1	1	Х	Х	Х	х	Х	х	Х	х	Х	х	х	Х	х	х	х	Stb
10	1	1	0	0	0	1	Х	х	х	1	0	Х	1	1	1	0	х	Х	1	0	1	1	0	1	Jalr
11	0	1	0	0	0	0	Х	х	х	0	0	х	х	х	1	0	х	х	1	0	0	0	х	х	Bz
12	1	0	0	0	0	0	Х	х	х	0	0	X	х	X	1	0	х	Х	1	0	0	0	х	х	Bnz
13	0	0	0	0	0	1	Х	х	х	Х	Х	0	0	0	1	0	0	1	1	0	0	1	1	0	Movi
14	0	0	0	0	0	1	Х	х	х	Х	0	0	0	0	1	0	0	1	1	0	1	0	1	0	Movhi
15	0	0	0	1	0	1	Х	х	х	х	х	х	1	0	х	Х	х	Х	х	х	Х	х	1	0	In
16	0	0	0	0	1	0	Х	х	х	х	х	х	х	х	х	Х	х	х	х	х	х	х	х	х	Out
1731	0	0	0	0	0	0	х	х	х	х	x	х	х	х	х	х	х	х	х	х	x	x	х	х	Nop

ROM OUT (binària i hexa)



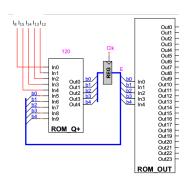
• Implementem les x amb '0'

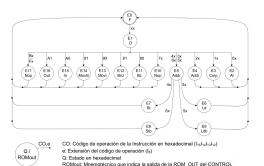
@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	MxN1	M×N0	MxF	F2	Œ	P0	Mx@D1	Mx@D0		
0	1	1	0	0	0	0	1	0	0	1	1	0	х	х	0	0	1	1	1	1	0	0	х	х	F	0xC260F0
1	0	0	0	0	0	0	0	х	х	х	1	0	х	х	0	0	1	0	1	1	0	0	х	х	D	0x0020B0
2	0	0	0	0	0	1	х	х	х	х	0	1	0	0	0	0	х	Х	0	х	х	х	0	0	Al	0x041000
3	0	0	0	0	0	1	х	х	х	х	0	1	0	0	0	1	х	Х	0	х	х	х	0	0	Cmp	0x041100
4	0	0	0	0	0	1	х	х	х	х	0	0	0	0	0	0	0	0	1	1	0	0	0	1	Addi	0x040031
5	0	0	0	0	0	0	0	х	х	х	0	0	х	х	0	0	0	0	1	1	0	0	х	х	Addr	0x000030
6	0	0	0	0	0	1	х	0	1	х	х	х	0	1	х	Х	х	Х	х	х	х	х	0	1	Ld	0x048401
7	0	0	1	0	0	0	х	0	1	х	х	х	x	х	х	х	х	х	х	х	х	х	x	x	St	0x208000
8	0	0	0	0	0	1	х	1	1	х	х	х	0	1	х	Х	х	Х	х	х	х	х	0	1	Ldb	0x058401
9	0	0	1	0	0	0	х	1	1	х	х	х	x	х	х	Х	х	Х	х	х	х	х	x	x	Stb	0x218000
10	1	1	0	0	0	1	х	х	х	1	0	х	1	1	1	0	х	Х	1	0	1	1	0	1	Jalr	0xC44E2D
11	0	1	0	0	0	0	х	х	х	0	0	х	х	х	1	0	х	х	1	0	0	0	х	х	Bz	0x400220
12	1	0	0	0	0	0	х	х	x	0	0	х	x	х	1	0	х	Х	1	0	0	0	x	x	Bnz	0x800220
13	0	0	0	0	0	1	х	х	х	х	х	0	0	0	1	0	0	1	1	0	0	1	1	0	Movi	0x040266
14	0	0	0	0	0	1	х	х	x	х	0	0	0	0	1	0	0	1	1	0	1	0	1	0	Movhi	0x04026A
15	0	0	0	1	0	1	х	х	х	х	х	х	1	0	х	Х	х	Х	х	х	х	х	1	0	In	0x140802
16	0	0	0	0	1	0	х	х	х	х	х	х	x	х	х	х	х	х	х	х	х	х	x	х	Out	0x080000
1731	0	0	0	0	0	0	х	х	х	х	х	х	х	х	х	х	х	х	х	х	х	х	х	х	Nop	0x000000

ROM Q+ (de l'estat següent)



- A partir de l'estat actual (k=5 bits), i de l'entrada (n=5 bits: bits I_{15} , I_{14} , I_{13} , I_{12} , I_{8} de la instrucció) calcula l'estat següent (k=5 bits)
 - 2^{n+k} paraules \times k bits/paraula = $2^{10} \times 5 = 5.120$ bits
- Representarem el contingut de la ROM Q+ de 3 formes equivalents





ROM Q+: contingut estats Fetch i Decode



- Tres possibles representacions del contingut
 - Simbòlica, taula de veritat compacta, "Logic Works"
 - La codificació de l'estat de la UC (5 bits) ocupa dos dígits hexadecimals
- Paraules corresponents als estats Fetch i Decode

Q	1	Q⁺	q ₄ q ₃ q ₂ q ₁ q ₀	I ₁₅ I ₁₄ I ₁₃ I ₁₂ I ₈	Q+ (Hexa)	# veces	Q ⁺ (Hexa)
F	Х	D	00000	XXXXX	01	32	01
D	AL	Al	00001	0000x	02	2	02
D	CMP	Cmp	00001	0 0 0 1 x	03	2	03
D	ADDI	Addi	00001	0010x	04	2	04
D	LD	Addr	00001	0 0 1 1 x	05	2	05
D	ST	Addr	00001	0100x	05	2	05
D	LDB	Addr	00001	0101x	05	2	05
D	STB	Addr	00001	0110x	05	2	05
D	JALR	Jalr	00001	0111x	0A	2	0A
D	BZ	Bz	00001	10000	0B	1	0B
D	BNZ	Bnz	00001	10001	0C	1	0C
D	MOVI	Movi	00001	10010	0D	1	0D
D	MOVHI	Movhi	00001	10011	0E	1	0E
D	IN	In	00001	10100	0F	1	0F
D	OUT	Out	00001	10101	10	1	10
D	ilegal	Nop	00001	1011x	11	2	11
D	ileyal	Nob	00001	11xxx	11	8	11
		-			4 □ > 4 ₱ 1	 4 ≣ → 	< ≣ > 3

ROM Q+: contingut estats de càlcul



• Paraules corresponents als estats de càlcul i modificació de l'estat

Q	1	Q⁺	q ₄ q ₃ q ₂ q ₁ q ₀	1 ₁₅ 1 ₁₄ 1 ₁₃ 1 ₁₂ 1 ₈	Q ⁺ (Hexa)	# veces	Q ⁺ (Hexa)
Al	Х	F	00010	xxxxx	00	32	00
Cmp	Х	F	00011	xxxxx	00	32	00
Addi	Х	F	00100	xxxxx	00	32	00
	1 (T.D.) O.D.)		00101	0000x	XX	2	00
Addr	! (LD+ST+ LDB+STB)	х	00101	0001x	xx	2	00
	TDD+21D)		00101	0010x	xx	2	00
Addr	LD	Ld	00101	0011x	06	2	06
Addr	ST	St	00101	0100x	07	2	07
Addr	LDB	Ldb	00101	0101x	08	2	80
Addr	STB	Stb	00101	0110x	09	2	09
Addr	! (LD+ST+	x	00101	0111x	XX	2	00
Addi	LDB+STB)	^	00101	1 x x x x	xx	16	00
Ld	Х	F	00110	xxxxx	00	32	00
St	Х	F	00111	xxxxx	00	32	00
Ldb	Х	F	01000	xxxxx	00	32	00
Stb	Х	F	01001	xxxxx	00	32	00
Jalr	Х	F	01010	xxxxx	00	32	00
Bz	х	F	01011	xxxxx	00	32	00
Bnz	х	F	01100	xxxxx	00	32	00
Movi	Х	F	01101	xxxxx	00	32	00
Movhi	х	F	01110	xxxxx	00	32	00
In	Х	F	01111	xxxxx	00	32	00
Out	X	F	10000	XXXXX	00	32	00

ROM Q+: contingut estats sense utilitzar



• Paraules corresponents a estats no utilitzats

Q	ı	Q⁺
Nop	x	F

q ₄ q ₃ q ₂ q ₁ q ₀	I ₁₅ I ₁₄ I ₁₃ I ₁₂ I ₈	Q+ (Hexa)
10001	xxxxx	00
1001x	$x \times x \times x$	00
101xx	$x \times x \times x$	00
1 1 x x x	$x \times x \times x$	00

# veces	Q + (Hexa)
32	00
64	00
128	00
256	00

ROM Q+: contingut format Logic Works



```
03.03
04 04
05.05
05.05
05 05
05.05
On Oa
06
0c
0d
0e
0£
11 11
11 11 11 11 11 11 11 11
00 00
00.00
06.06
07.07
08 08
09 09
00.00
```

ROM Q+: contingut format Logic Works



- Contingut de la ROM posició a posició
 - La primera dada correspon a la posició (fila) 0
 - Les següents dades omplen la ROM en posicions consecutives
- Els espais en blanc i els salts de línia actuen com a separadors
 - Hem utilitzat els salts de línia per fer el contingut més llegible
- Descripció del contingut:
 - Primera línia: 32 transicions des de Fetch a Decode
 - Següents 16 línies: 32 transicions des de *Decode* a un node de càlcul
 - Següent línia: 32 transicions des de l'estat Al a Fetch
 - Següent línia: 32 transicions des de l'estat Cmp a Fetch
 - Següent línia: 32 transicions des de l'estat Addi a Fetch
 - Següents 9 línies: 32 transicions des de l'estat Addr a Fetch o a Ld, Ldb, St, Stb.
 - Següent línia: 32 transicions des de l'estat Ld a Fetch
 - ...



ROM Q+: contingut format Logic Works



```
02 02
04 04
05 05
05 05
05 05
05 05
0a 0a
     Decode → Primer cicle càlcul
0b
0e
11 11
00 00
00 00
00 00
06 06
          Addr → Ld / Ldb / St /Stb
08 08
09 09
00 00
```

Índex



- Introducció
- Esquema lògic de la Unitat de Control
- Exercicis
- Conclusions

Exemple: contingut ROM Q+



- Exemple: contingut de l'adreça 0xAC de la ROM Q+?
 - Les adreces de la ROM Q+ tenen 10 bits:
 - Els 5 de més pes són l'estat actual
 - Els 5 de menys pes són els bits 15,14,13,12 i 8 de la instrucció
 - $0xAC \rightsquigarrow (10101100)_2 \rightsquigarrow (0010101100)_2 \rightsquigarrow (00101\ 01100)_2$
 - Estat actual = $00101_2 \rightsquigarrow 5_{10} \rightsquigarrow Addr$
 - Codi d'operació = $01100_2 \rightsquigarrow (0110\ 0)_2 \rightsquigarrow STB$
 - Mirant el graf d'estats de la UC, si estem a l'estat Addr i la instrucció és STB, l'estat futur és Stb
 - Com el codi de l'estat Stb és 9, el contingut de l'adreça 0xAC de la ROM Q+ és $01001_2=09_{16}=0x09$

Exemple: contingut ROM Q+



- Exemple: quines adreces de la ROM Q+ contenen l'estat Addr?
 - Mirant el graf d'estats de la UC, veiem que l'estat Addr és l'estat futur de l'estat D per a les instruccions LD, ST, LDB, STB
 - D és codifica amb $1 = (00001)_2$
 - Els codis d'operació de LD, ST, LDB, STB són 0011₂, 0100₂, 0101₂, 0110₂ respectivament
 - Aquestes instruccions no tenen bit extra de codi d'operació
 - Adreces ROM Q+: 5 bits codificació de l'estat + 4 bits codi d'operació + 1 bit codi d'operació extra
 - Adreces involucrades:
 - $(00001\ 0011\ x)_2 \rightsquigarrow (00\ 0010\ 011x)_2 \rightsquigarrow 0x026\ i\ 0x027$
 - $(00001\ 0100\ x)_2 \rightsquigarrow (00\ 0010\ 100x)_2 \rightsquigarrow 0x028\ i\ 0x029$
 - $(00001\ 0101\ x)_2 \rightsquigarrow (00\ 0010\ 101x)_2 \rightsquigarrow 0x02A \ i\ 0x02B$
 - $(00001\ 0110\ x)_2 \rightsquigarrow (00\ 0010\ 110x)_2 \rightsquigarrow 0x02C\ i\ 0x02D$

Exercicis: contingut ROM Q+



• Indiqueu el contingut de les adreces 0xC i 0x26?

Quina(es) adreces de la ROM Q+ porten a l'estat In?

Quina(es) adreces de la ROM Q+ porten a l'estat Al?

Exercicis: contingut ROM Q+



- Indiqueu el contingut de les adreces 0xC i 0x26?
 - $0xC = 0000000110_2 = (00000\ 0110\ 0)_2 \implies$ Correspon a la transició Fetch (00000) amb codi d'operació Stb (0110x) per tant, ha d'anar a parar a Decode (00001) \implies el contingut és $00001_2 = 0x01$
 - $0x26 = 0000100110_2 = (00001\ 0011\ 0)_2 \Longrightarrow Correspon$ a la transició Decode (00001) amb codi d'operació Ld (0011x) per tant, ha d'anar a parar a Addr (00101) \Longrightarrow el contingut és $00101_2 = 0x05$
- Quina(es) adreces de la ROM Q+ porten a l'estat In?
 - Arribem a In des de Decode (00001) amb codi d'operació 10100 per tant, l'adreça (00001 1010 0) $_2=0000110100_2=0x034$
- Quina(es) adreces de la ROM Q+ porten a l'estat Al?
 - Arribem a Al des de Decode (00001) amb codi d'operació 0000x per tant, les adreces (00001 0000 x)₂ = 000010000x₂ = 0x020 i 0x021

Exercici: contingut ROM OUT



- Indiqueu el contingut d'un seguit de files/columnes/interseccions de la ROM OUT
 - Posseu x sempre que sigui possible

@ROM	Bz	M×F	WrD	R@/PC	Pc/Rx	LdIR	Ry/N	Estado
3								Cmp
5								Addr
9								Stb
12								Bnz

Exercici: contingut ROM OUT



- Indiqueu el contingut d'un seguit de files/columnes/interseccions de la ROM OUT
 - Posseu x sempre que sigui possible

@ROM	Bz	M×F	WrD	R@/PC	Pc/Rx	LdIR	Ry/N	Estado
3	0	0	1	х	0	х	1	Cmp
5	0	1	0	х	0	0	0	Addr
9	0	х	0	1	х	х	х	Stb
12	0	1	0	х	0	х	х	Bnz

Exercici: afegir una nova instrucció al LM



- Exercici típic: afegir una nova instrucció al LM
 - De moment, sense modificar hardware de la UC ni de la UP
 - Més endavant ens permetrem modificar el hardware
- Haurem de modificar el contingut de la ROM Q+ i de la ROM OUT
 - Afegir nous estats al graf d'estats de la UC
 - Definir les seves paraules de control
- S'assumeix que la resta d'instruccions del LM han de continuar funcionant com fins ara

Exemple: afegir instrucció LBZ



- Long Branch on Zero: permetrà fer un salt a una direcció absoluta
 - Sintaxi: LBZ Ra, Rb
 - Codificació: 1111 aaa bbb xxxxxx
 - Semàntica: if (Ra==0) PC \leftarrow Rb; else PC \leftarrow PC + 2;
- Quants nodes del graf d'estats necessita LBZ?
 - Fetch i Decode
 - No podem treure res del que s'està fent a aquests estats perquè afectaria a la resta d'instruccions
 - A Fetch ja es farà $PC \leftarrow PC + 2$
 - Cicles de càlcul?
 - Com a mínim 2 perquè necessitem la ALU per per a avaluar Ra==0 i per a poder fer arribar Rb a PC a través de R@

Exemple: afegir instrucció LBZ

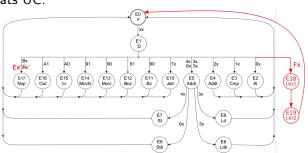


- Estratègia d'implementació: Imitarem BZ
 - Guardarem a R@ la direcció de salt Rb
 - Al cicle següent avaluarem Ra==0 deixant passar RX per la ALU; si z=1 actualitzarem PC amb R@
- Calen dos estats per a completar l'execució:
 - Lbz1: R@ ← RY
 - Lbz2: if (RX == 0) PC \leftarrow R0;
- Modificacions a les ROM's:
 - ROM Q+: reflectir les noves transicions
 - ROM OUT: per generar les paraules de control

Exemple: afegir instrucció LBZ v1.0



• Graf d'estats UC:



- Modificacions a la ROM Q+
 - ROM_Q+[00001 1111 x] = 0x12, ROM_Q+[10010 1111x] = 0x13 i ROM_Q+[10011 xxxxx] = 0x00
- Modificacions a la ROM OUT

18 0 0 0 0 0 0 0 0 0 0 x x x x 1 x x 1 0 x x 1 0 0 1 x x 1 0 0 0 0	@ROM																									
19 0 1 0 0 0 0 x x x 0 0 x x x 1 0 x x 1 0 0 0 x x	18	0	0	0	0	0	0	0	x	X	х	х	1	X	х	1	0	X	х	1	0	0	1	X	X	
	19	0	1	0	0	0	0	x	x	x	0	0	X	x	x	1	0	х	x	1	0	0	0	x	x	

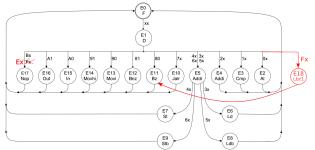
Acciones asociadas al estado (en lenguaje de transferencia de registros)

R® + RY
if (RX==0) PC + R®

Exemple: afegir instrucció LBZ v2.0



- Els estats Lbz2 i Bz són idèntics!!!
 - Des de Lbz1 podem passar a l'estat Bz
- Graf d'estats UC:



- Modificacions a la ROM Q+
 - $ROM_Q+[00001\ 1111\ x] = 0x12\ i\ ROM_Q+[10010\ 1111x] = 0x0B$
- Modificacions a la ROM OUT

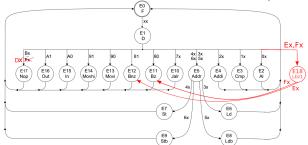


Acciones asociadas al estado (en lenguaje de transferencia de registros)

Exemple: afegir instrucció LBNZ



- Afegir també instrucció LBNZ (Long Branch on No Zero)
 - Anàloga a LBZ, amb codi d'operació Ex
- Graf d'estats UC:
 - Reutilitzarem estat Lbz
 - Transicionarà a node Bz o Bnz en funció del codi d'operació



- Modificacions a la ROM Q+ (a més de les anteriors)
 - ROM Q+ $[00001\ 1110\ x] = 0x12\ i\ ROM\ Q+[10010\ 1110x] = 0x0C$
- Modificacions a la ROM OUT (a més de les anteriors)
 - Cap

Exercicis a entregar a Atenea



- Enunciat disponible a Atenea
 - https://atenea.upc.edu/pluginfile.php/3603451/mod_assign/introattachment/0/Tema%2013%20-%20Exercicis%20en%20paper.pdf?forcedownload=1
- Entrega a Atenea fins el dijous 17/12
 - Format PDF
 - Heu d'indicar cicle a cicle quina és l'evolució de la paraula de control a l'executar un determinat programa
 - Heu de justificar la preeminência de l'arquitectura von Neumann sobre la Harvard
 - Amb el tema 13c acabareu de veure els arguments

Índex



- Introducció
- Esquema lògic de la Unitat de Control
- Exercicis
- Conclusions

Conclusions



- Hem implementat la Unitat de Control del computador von Neumann
 - CLS amb 18 estats
 - Interpreta instruccions SISA i genera els senyals de control adients
 - Cal saber deduir contingut de les ROM's de l'estat següent i de sortides
- Podem crear noves instruccions SISA modificant les ROMs Q+ i OUT
- A la propera classe:
 - Especificarem la darrera instrucció de LM SISA: JALR
 - Estudiarem la temporalitat del computador Von Neumann
 - Veurem exemples més complexes d'afegir noves instruccions
- Contesteu el questionari ET13b i l'exercici 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] Pentium FDIV bug, [Online]. Available: https://en.wikipedia.org/wiki/Pentium_FDIV_bug#Affected_models.
- [2] Meltdown (security vulnerability), [Online]. Available: https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability).
- [3] Microarchitectural Data Sampling, [Online]. Available: https://en.wikipedia.org/wiki/Microarchitectural_Data_Sampling.
- [4] KB4497165: Intel microcode updates, [Online]. Available: https://support.microsoft.com/en-ca/help/4497165/intel-microcode-updates.

Introducció als Computadors

Tema 13: Computador SISC von Neumann http://personals.ac.upc.edu/enricm/Docencia/IC/IC13b.pdf

Enric Morancho (enricm@ac.upc.edu)

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



2020-21, 1^{er} quad.

Presentació publicada sota Ilicència Creative Commons 4.0 @ (1)

