Introducció als Computadors

Tema 9: Entrada/Sortida

http://personals.ac.upc.edu/enricm/Docencia/IC/IC9.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.

Presentació publicada sota llicència Creative Commons 4.0

E/S per Enquesta vs. Interrupcions

- Analogia amb mitjans de comunicació encara existents

[1]

[2]

Enquesta-Espera activa
(Polling-Busy waiting)

Ho veurem a IC

Interrupcions
(Interrrupts)

Ho veureu a EC

Índex

Full de ruta

	Tema							
	7	8	9	10	11	12	13	14
Unitat de Control	UCE	UCE	UCE	UCG	UCG	UCG	UCG	UCG
Unitat de Procés	UPE	UPG	UPG	UPG	UPG	UPG	UPG	UPG
Entrada/Sortida	-	-	IO	IO	IO	IO	IO	IO
Memòria RAM	-	-	-	-	MEM	MEM	MEM	MEM
Harvard unicicle	-	-	-	-	-	✓	-	-
Harvard multicicle	-	-	-	-	-	✓	-	-
Von Neumann	-	-	-	-	-	-	✓	✓
Lleng. *assembler*	-	-	-	✓	✓	✓	✓	✓

- Connectarem perifèrics al nostre computador
  - Dispositius mitjançant els quals es comunica amb l'exterior
    - teclat, impressora, ...
  - Els perifèrics realitzen operacions d'entrada/sortida (E/S)
    - Input/Output (I/O)
- Categorització dels perifèrics:
  - Els perifèrics d'entrada enviaran dades a través del bus RD-IN
    - Exemple: teclat
  - Els perifèrics de sortida rebran les dades a través del bus WR-OUT
    - Exemple: impressora
- La comunicació entre UPG i perifèrics serà **asíncrona**
  - Els perifèrics **no** poden compartir senyal de rellotge amb el PPE
    - Diferents velocitats de procés
    - Distància entre PPE i perifèric

- La documentació del tema a Atenea presenta el procés de creació del sistema d'E/S amb diversos passos intermedis
- A aquestes transparències s'explica el procés més directament
- Assumirem que connectarem un "teclat" i una "impressora"
  - Però serà generalitzable a altres dispositius
  - Esquema lògic disponible al simulador Logic Works
- A efectes pràctics, heu de saber com fer que la UPG pugui comunicar-se amb els perifèrics d'E/S
  - 3 nodes al graf d'estats de la UC

Índex

ports d'E/S

comunicar-se amb un perifèric voldrà dir
- Per a la UPG, un perifèric d'E/S serà un conjunt de registres
  - Llegir/escriure sobre aquests registres la comunicarà amb el perifèric
  - Aquests registres rebran el nom de *ports* d'E/S
- Tots aquests registres s'agruparan a un dispositiu controlador (K)
  - K estarà connectat als busos RD-IN i WR-OUT
  - Farà d'intermediari entre la UPG i els perifèrics
    - No veurem com s'implementa cada perifèric
  - La UPG llegirà/escriurà els *ports* amb les accions `IN/OUT`
- Potencialment, podrem connectar diversos perifèrics a la UPG i cada perifèric podrà tenir associats varis *ports*
- Caldrà identificar quin *port* es vol llegir/escriure
  - Cada *port* tindrà associat un nombre natural de 8 bits
    - 2^8=256 *ports* d'entrada i 256 *ports* de sortida
    - Aquest identificador l'anomenarem ADDR-IO
  - ADDR-IO formarà part de la paraula de control generada per la UC

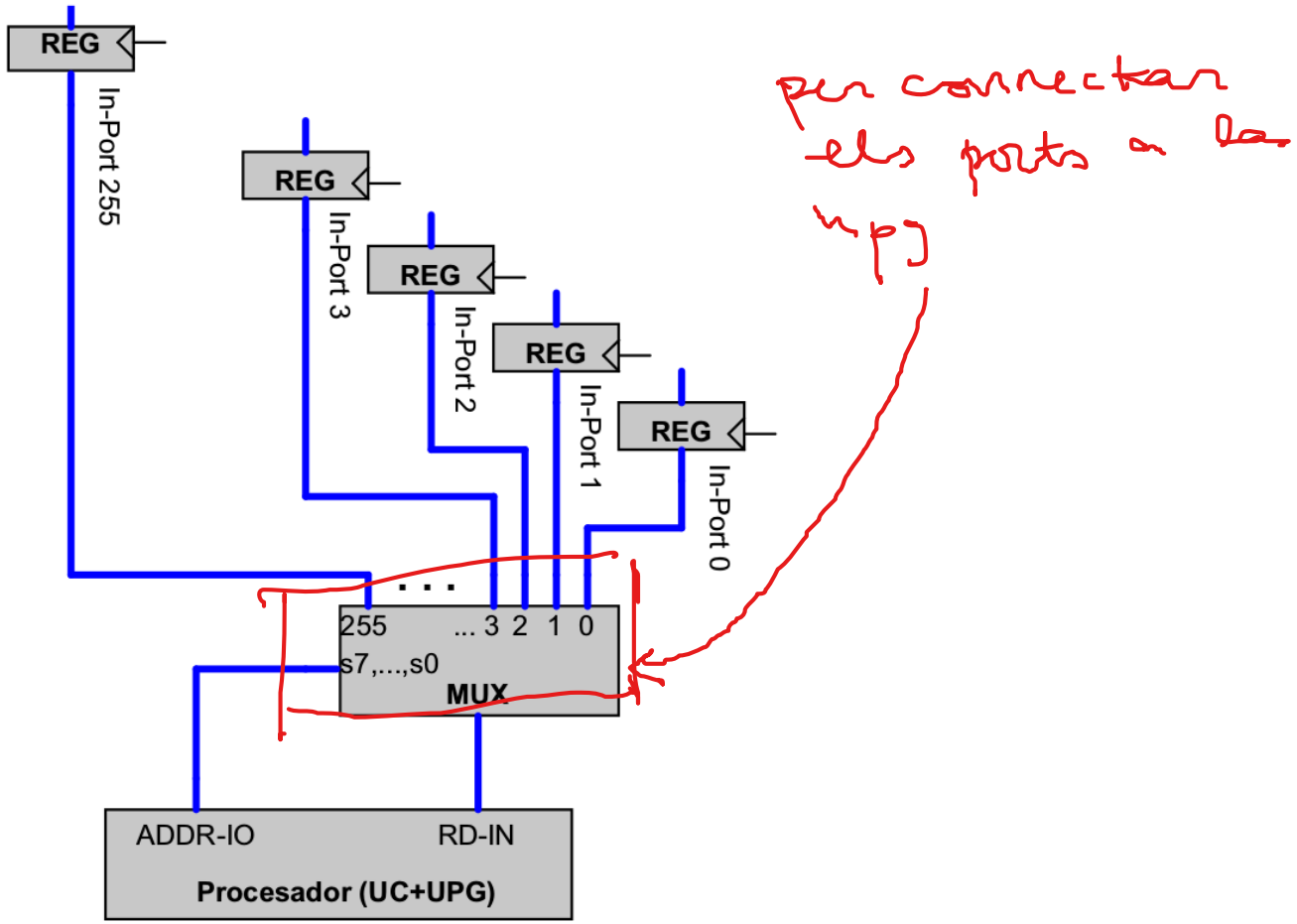
comunicació a través del controlador intermediari entre CPU i perifèric

Reformulació de les accions IN i OUT

- Reformulació acció `IN`
  - Llegeix un dels 256 *ports* d'entrada i, al final del cicle, escriu el seu valor a un registre
  - Exemple:
    - `IN R1, 5`
    - Llegeix el *port* d'entrada número 5 i guarda el seu contingut a `R1`
- Reformulació acció `OUT`
  - Escriu a un dels 256 *ports* de sortida el valor d'un dels registres del banc de registres de la UPG
  - Exemple:
    - `OUT 4, R2`
    - Escriu el valor de `R2` al *port* de sortida número 4
  - La UC generarà un nou bit de control: Wr-Out
    - Pels busos WR-OUT i ADDR-IO sempre hi ha algun valor
    - Wr-Out indica si en aquest cicle realment s'està fent una acció `OUT`
    - És a dir, si aquest cicle el valor dels busos WR-OUT i ADDR-IO és vàlid
    - Anàleg al bit de control WrD del banc de registres

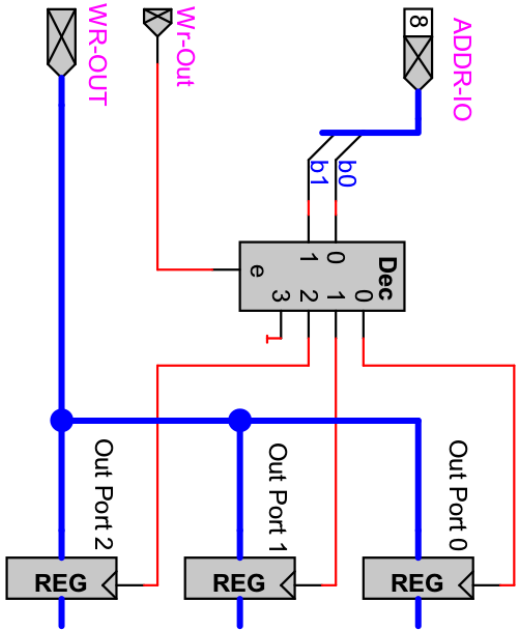
Connectant *ports* d'entrada amb UPG

- Un multiplexor de busos 256-1 seleccionarà el contingut de quin *port* d'entrada s'encaminarà cap al bus RD-IN
  - ADDR-IO actua com a senyal de selecció del multiplexor 256-1



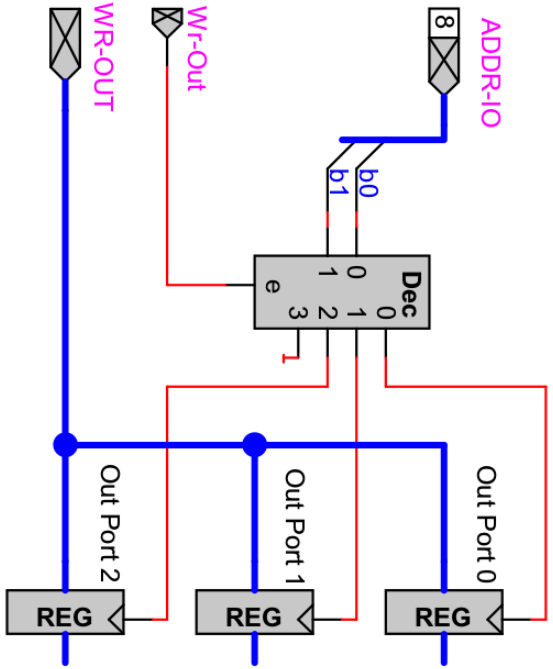
Connectant UPG amb *ports* de sortida

- El bus WR-OUT s'encamina a l'entrada de tots els *ports* de sortida
- Un descodificador amb senyal d'*enable* genera el senyal de rellotge sobre els *ports* de sortida
  - El senyal Wr-Out s'utilitza com a entrada *enable* del descodificador
    - Anàleg al senyal WrD al REGFILE
    - Si *enable* val "0", cap *port* de sortida es modifica
    - Si *enable* val "1", només es modifica el *port* ADDR-IO
- Exemple: implementació amb 3 *ports* de sortida
  - Dels 8 bits de ADDR-IO, s'ignoren els 6 bits de més pes



Observacions

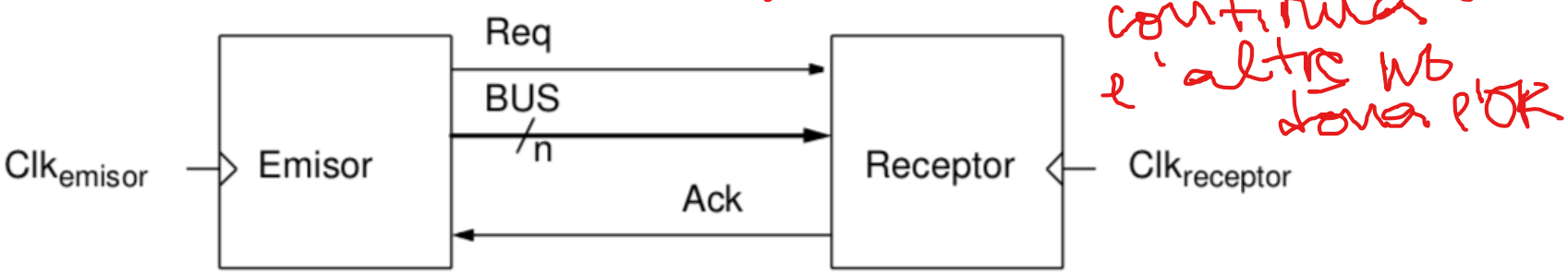
- Hem creat un espai d'adreces d'E/S
  - 0..255 *ports* d'entrada i 0..255 *ports* de sortida
- Quan s'escrigui a un *port* de sortida, cal que el valor al bus WR-OUT sigui estable abans que el senyal de control Wr-Out passi a "1"
  - Altrament, podríem escriure un valor incorrecte al *port* de sortida
  - De moment no ho solucionem, ho farem a un tema posterior



Índex

Comunicació asíncrona

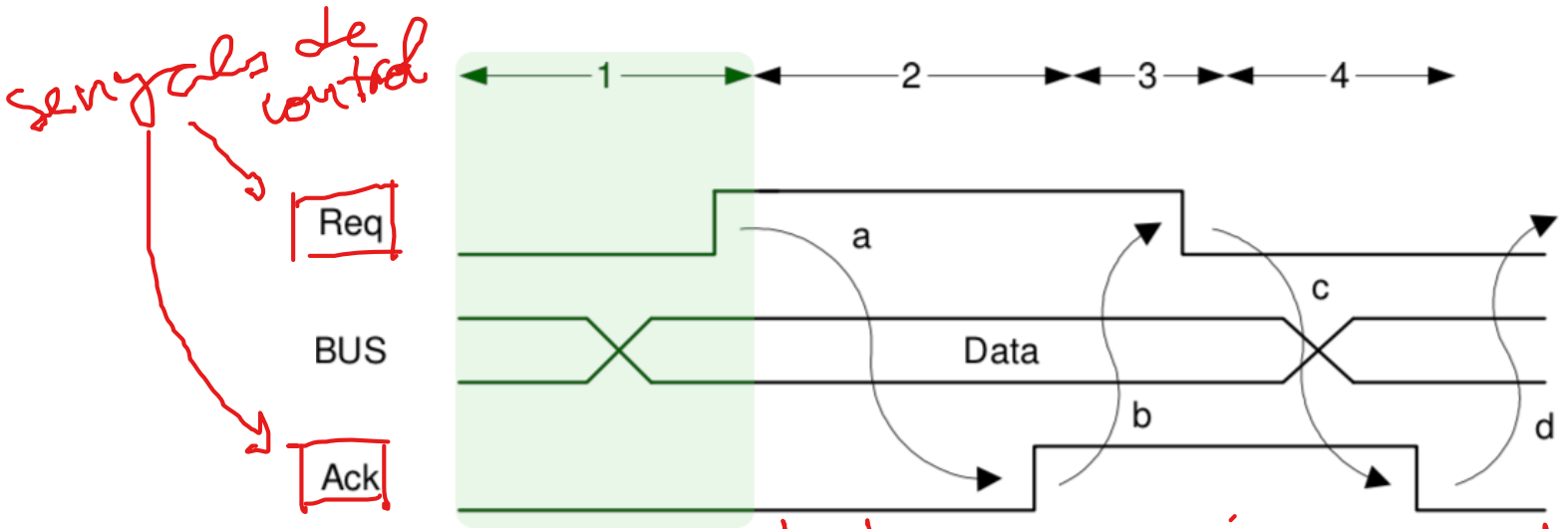
- Comunicar un emissor i un receptor sense compartir senyal de rellotge
  - Cal utilitzar un protocol de comunicació asíncron
    - **Four-phase handshake**
- Senyals que hi intervendran:
  - De control: *request* i *acknowledgment* d'un bit cadascun
    - Estat inicial: senyals *request* i *acknowledgment* a "0"
  - De dades: BUS: bus de *n* bits



- Assumim comunicació iniciada per l'emissor
  - Típic a perifèrics d'entrada com ara el teclat
  - El teclat fa el paper d'emissor i la UPG fa el paper de receptor
  - El senyal *request* indica que hi ha una dada disponible

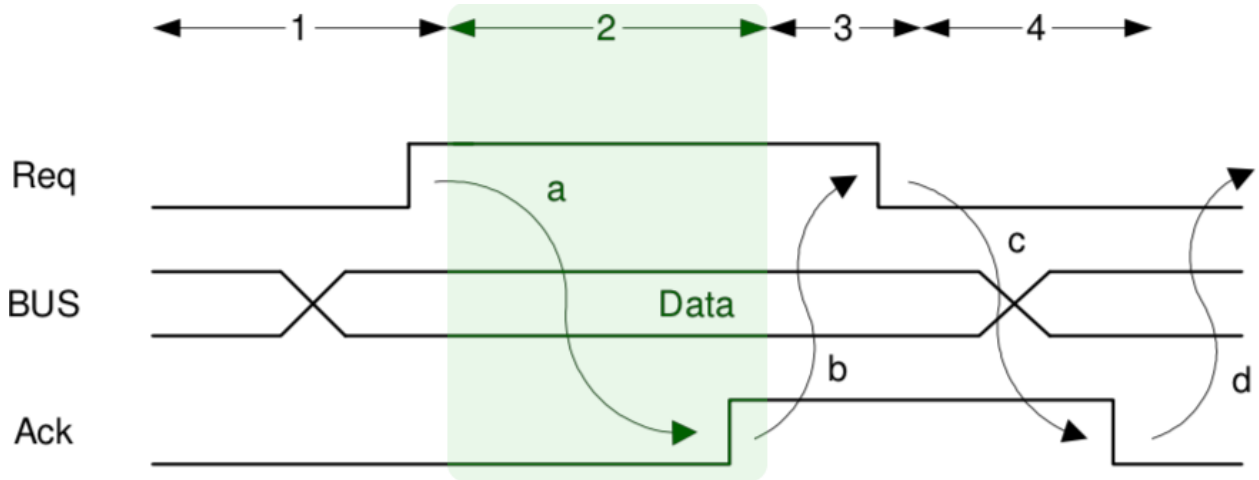
Protocol *four-phase handshake*: fase 1

- Des de l'estat inicial (*request* =*acknowledgment* ="0")
  - L'emissor posa la dada a transmetre al bus de dades i la manté estable
  - L'emissor indica al receptor que hi ha una dada disponible
    - Posa el senyal *request* a "1" i el manté estable
- És clau l'ordre d'activació dels senyals
  - Altrament, el receptor podria llegir una dada que encara no és estable



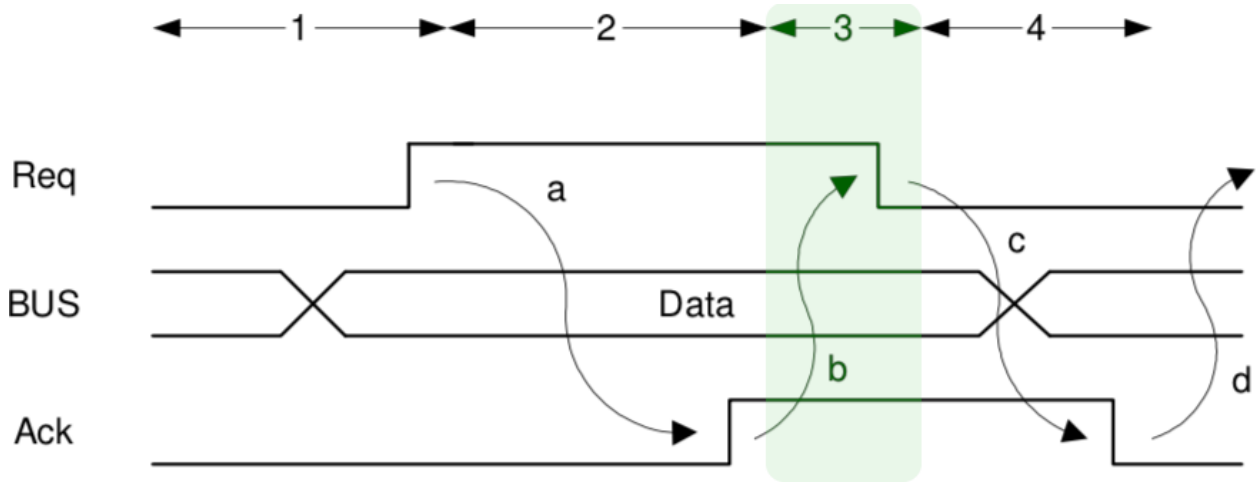
Protocol *four-phase handshake*: fase 2

- Quan el receptor observa el canvi al senyal *request*
  - El receptor llegeix la dada del bus de dades
  - El receptor indica a l'emissor que ja l'ha llegida
    - Posa a el senyal *acknowledgment* a "1" i el manté estable
    - A partir d'ara, l'emissor pot retirar la dada del bus
- És clau l'ordre d'activació dels senyals
  - Altrament, l'emissor podria retirar la dada abans de ser llegida



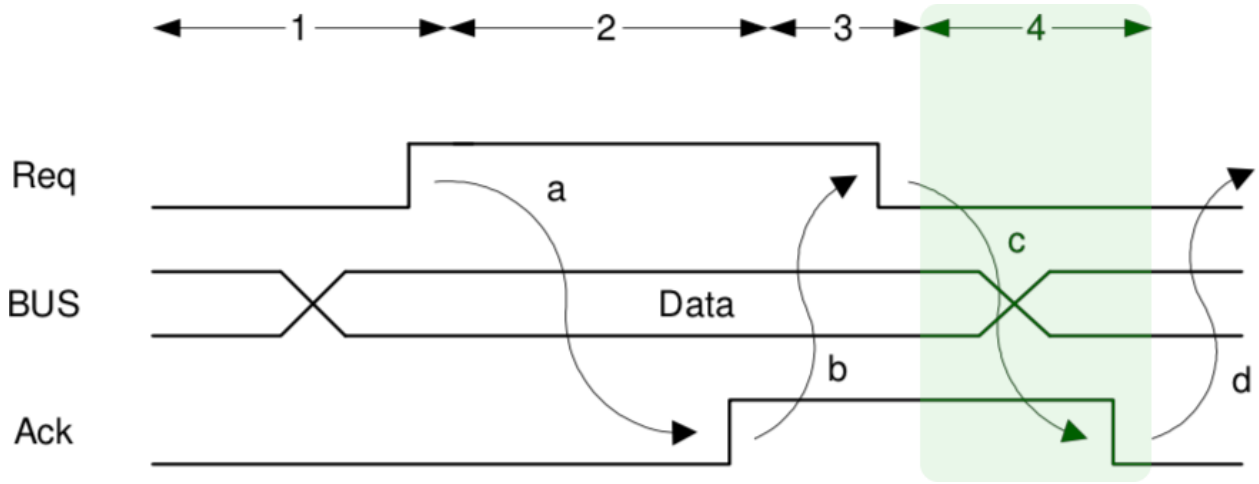
Protocol *four-phase handshake*: fase 3

- Quan l'emissor observa el canvi al senyal *acknowledgment*
  - L'emissor torna al seu estat inicial
    - El senyal *request* torna a valer "0" i el manté estable
    - *Return to zero* (RZ)
    - Ja no cal mantenir la dada estable al bus



Protocol *four-phase handshake*: fase 4

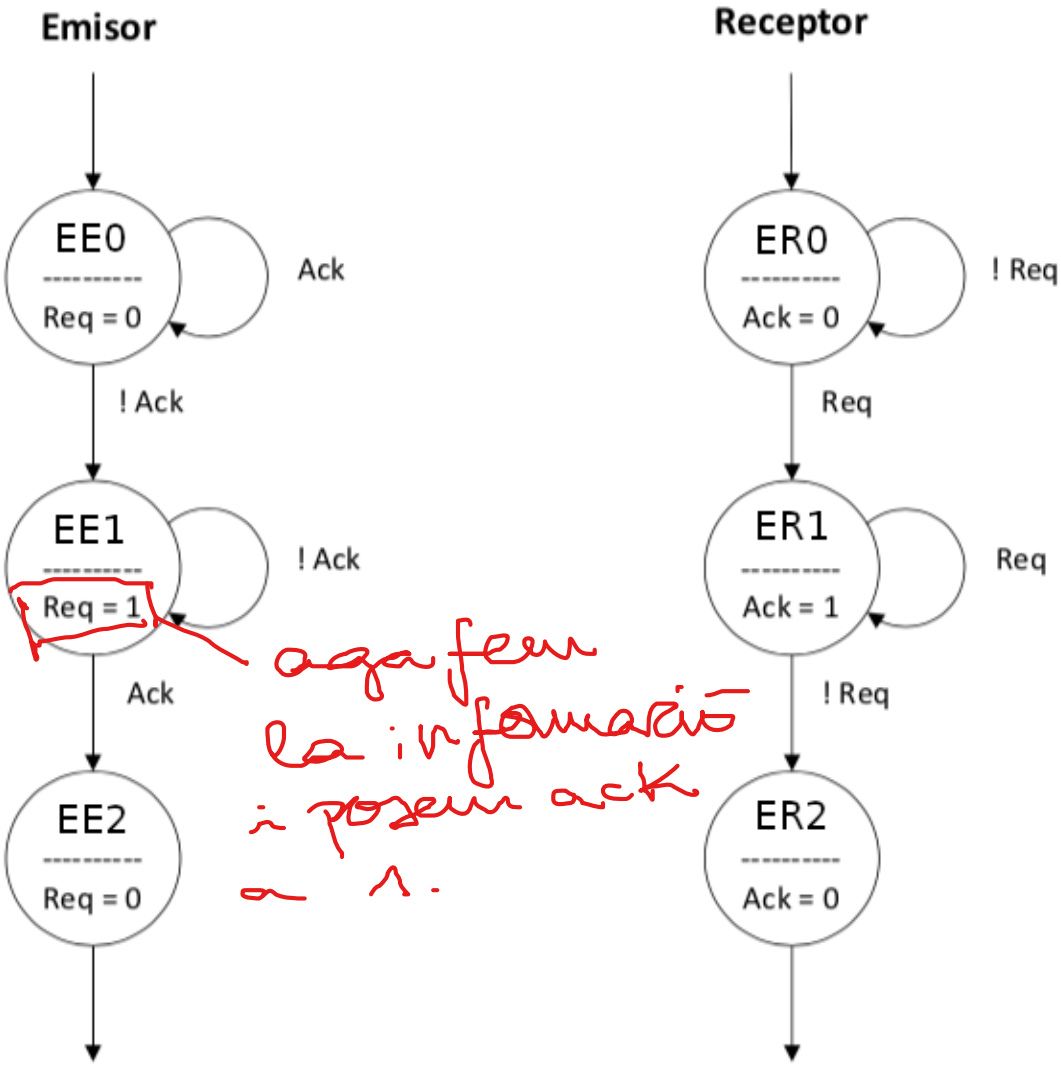
- Quan el receptor observa el canvi al senyal *request*
  - El receptor torna al seu estat inicial
    - El senyal *acknowledgment* torna a valer "0" i el manté estable
    - *Return to zero* (RZ)
- A partir d'ara, l'emissor pot iniciar una nova transferència



Emissor ja té el request a 0 i acaba la comunicació

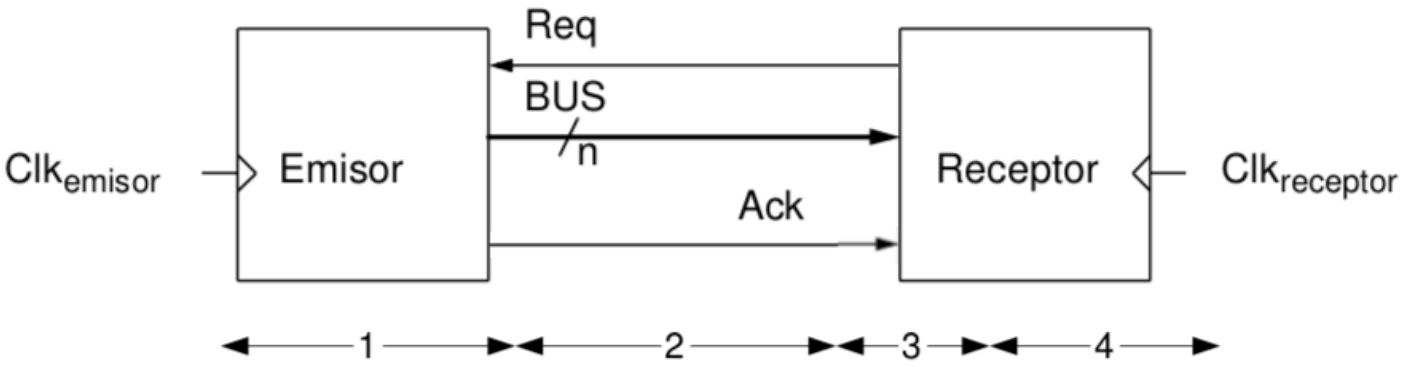
Protocol 4 fases: diagrama d'estats

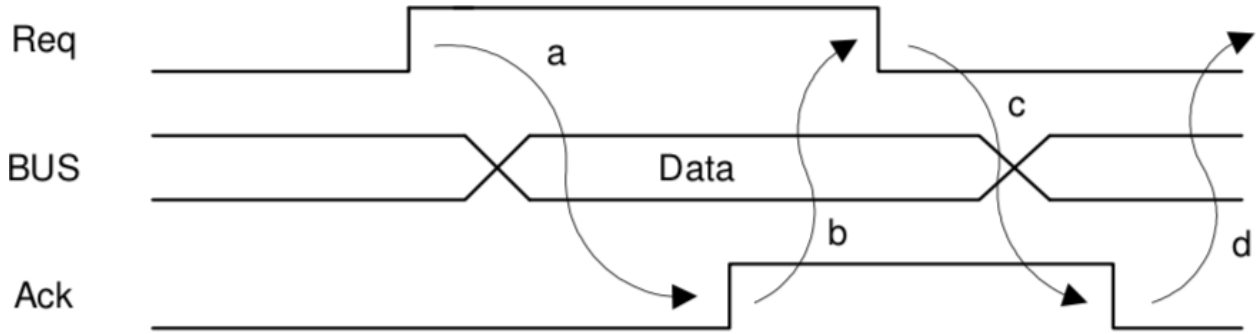
- Fragments dels grafs d'estats de l'emissor i del receptor per fer una transferència



Protocol 4 fases: observacions

- Funciona sigui quina sigui la velocitat de l'emissor i del receptor
- També es pot adaptar per a perifèrics de sortida (ex.: impressora)
  - Cal canviar el sentit d'enviament dels senyals de control i els papers
  - El senyal *request* indica que la impressora està disponible per acceptar nous caràcters
  - La UPG fa el paper d'emissor i la impressora de receptor





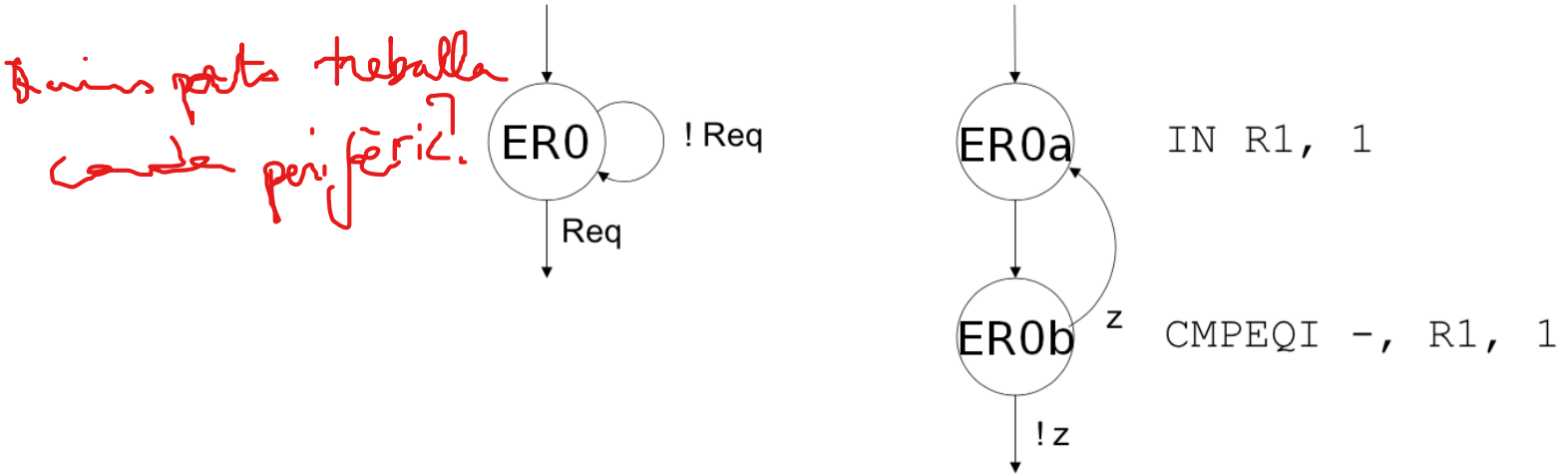
Protocol 4 fases a la UPG: senyals control

Tindrem ports que fan de req i ports que fan d'ack.

- Com representem els senyals *request* i *acknowledgment*?
  - Opció 1: afegir-les com a senyals de control a la UC
    - Calen dos senyals per a cada perifèric!
    - Necessitaríem UC's amb moltes entrades de control
  - Opció 2: **incorporar aquests senyals a un *port* del perifèric**
    - Tindrem *ports* d'entrada de dades i *ports* d'entrada d'estat
      Escrits pel perifèric i llegits per la UPG
    - Tindrem *ports* de sortida de dades i *ports* de sortida de control
      Escrits per la UPG i llegits pel perifèric
- Triarem l'opció 2:
  - La UC tindrà únicament un bit de entrada (z)
  - Els grafs d'estats de la UC seran molt senzills
    - Com a molt, cada estat tindrà dos possibles estats futurs
  - El preu és que seguint l'opció 2 els grafs d'estats de la UC poden necessitar més cicles que amb l'opció 1

Protocol 4 fases a la UPG: lectura

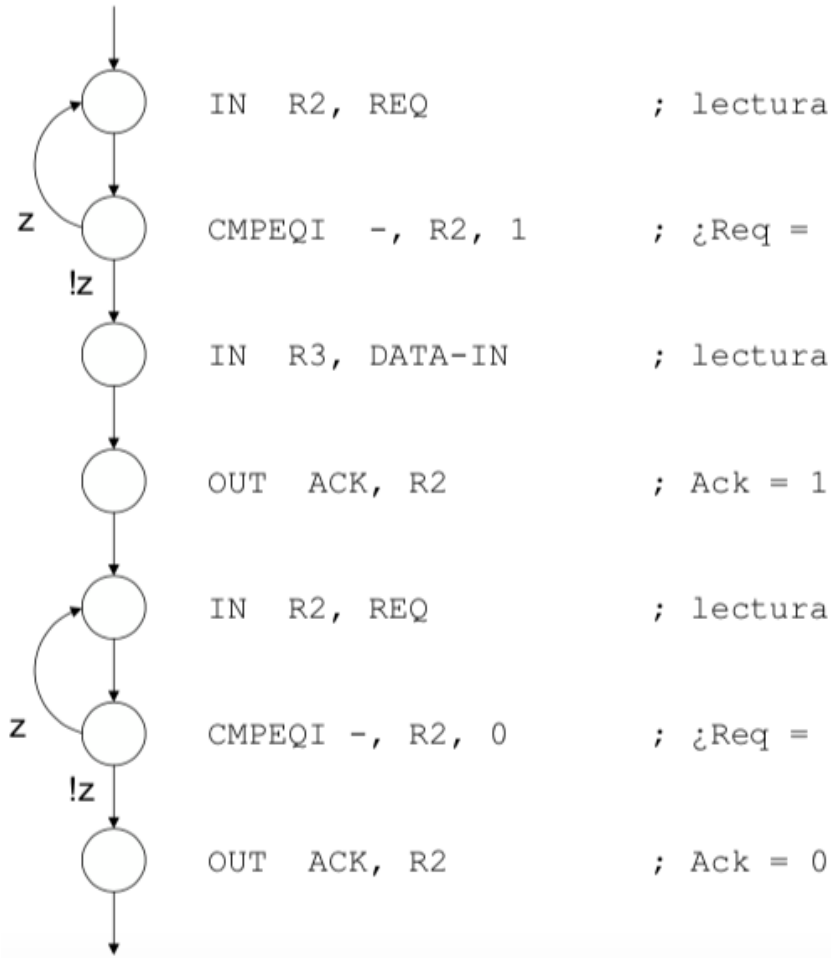
- El perifèric és l'emissor i la UPG el receptor
- Implementarà la part del receptor del protocol
- Com esperar que el perifèric activi el seu senyal *request*?
  - Cal saber en quin *port* estarà mapejat aquest senyal
    - Ens especifiquen que estarà en el bit de menys pes del *port* 1
    - La resta de bits del *port* ens asseguren que sempre valdran "0"
- Representació com a accions de la UPG:



- La UPG està fent una espera activa (*polling*)
  - La UPG continuament comprova el valor de *request* fins que valgui "1"

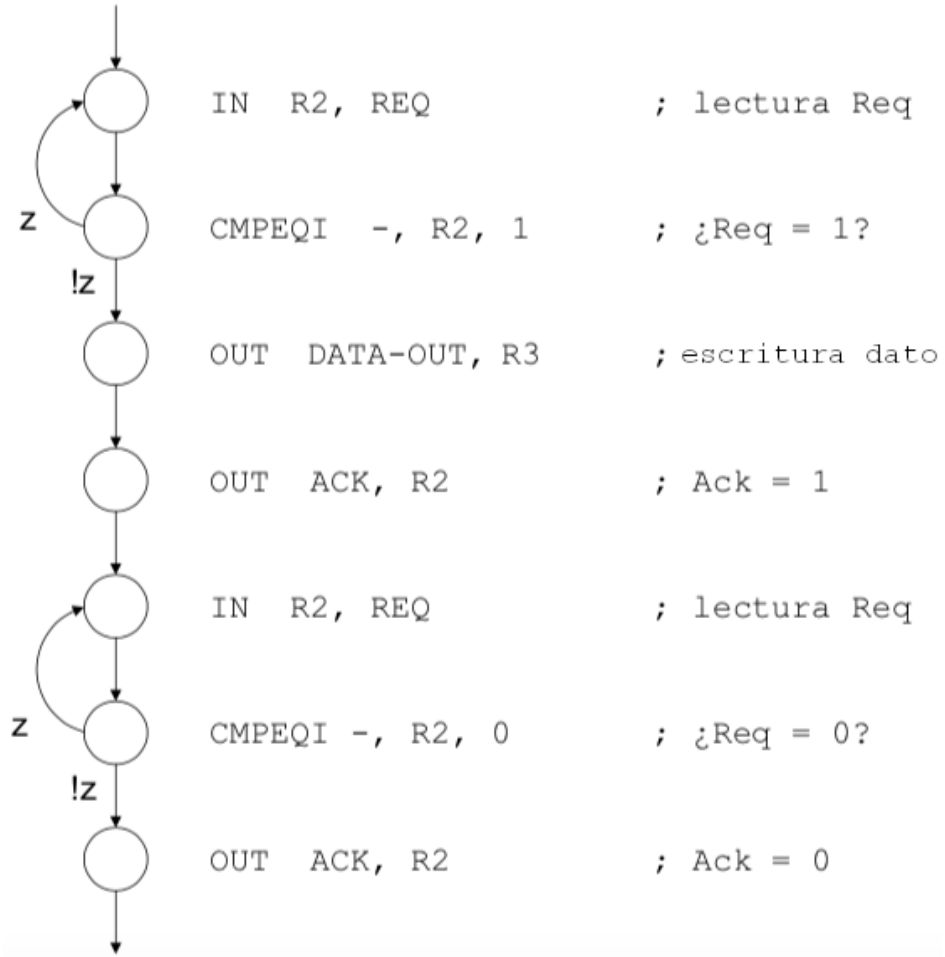
Protocol 4 fases a la UPG: lectura

- REQ, DATA-IN i ACK són els identificadors dels *ports* que emmagatzemen senyal de *request* (*port* d'estat), busos de dades i senyal de *acknowledgment* (*port* de control)



Protocol 4 fases a la UPG: escriptura

- El perifèric és el receptor i la UPG l'emissor
- DATA-OUT és l'identificador del *port* que emmagatza bus de dades



Protocol 4 fases a la UPG: observacions

- Llegir/escriure una dada requereix 7 nodes del graf d'estats de la UC
  - En són molts!
- A continuació ho reduirem a 3 nodes
  - El controlador K implementarà els 4 darrers nodes
  - La lectura d'un *port* de dades tindrà efecte lateral en un *port* d'estat
    - La lectura d'un *port* de dades modificarà un altre *port*
  - Anàlogament amb l'escriptura sobre el *port* de dades d'un perifèric
    - Modificarà el *port* d'estat del perifèric

Índex

Efecte lateral accedint als *ports* de dades

De forma automàtica quan llegim els ports de dades es proposa una escritura d'ACK i

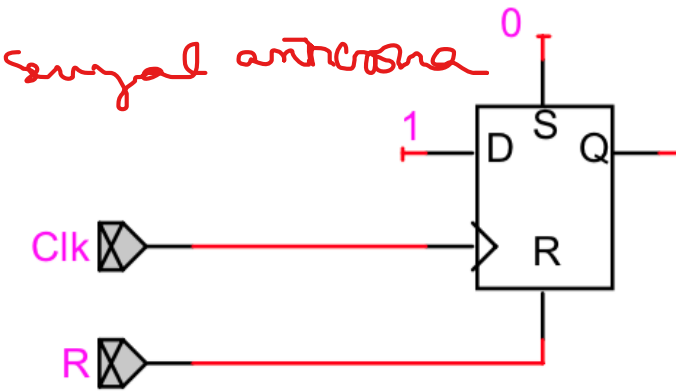
- K implementarà les 4 darreres accions que hauria de realitzar la UPG
- Fins ara la UPG tots els cicles estava llegint el *port* ADDR-IO
  - No era un problema perquè les lectures no tenien cap efecte lateral
  - Els bits de control In-ALU i WrD acabaven descartant aquesta dada
- Com ara les lectures poden tenir efecte lateral...
  - La UC afegeix el senyal del control Rd-In per indicar els cicles en que realment s'està fent una acció `IN`
- La lectura del *port* de dades d'un perifèric provocarà que el bit de control *acknowledgment* i el bit d'estat *request* tornin a valer "0"
  - Anàlogament amb l'escriptura sobre el *port* de dades d'un perifèric
- K haurà de detectar quan es llegeix/escriu sobre el *port* de dades
  - Monitoritzarà els senyals Rd-In, Wr-Out i ADDR-IO

Implementació *ports* de control

- Utilitza biestable D activat per flanc ascendent i amb entrades asíncrones de posada a "0" (R) i a "1" (S)
  - Quan es produeixi flanc ascendent a Clk, el valor a D passa a Q
  - Si S val 1, Q es posa a "0" immediatament
  - Si R val 1, Q es posa a "1" immediatament
- Implementació del *port* de control
  - Bit 0:
    - Aquest biestable definirà el bit 0 del *port* de control
    - Quan hi hagi flanc ascendent a Clk, Q valdrà "1" (*request* = "1")
    - **L'entrada R el posarà a "0" asíncronament (*request* = "0")**
    - Mai el posarem a "1" asíncronament
  - La resta de bits del *port* valdran "0"

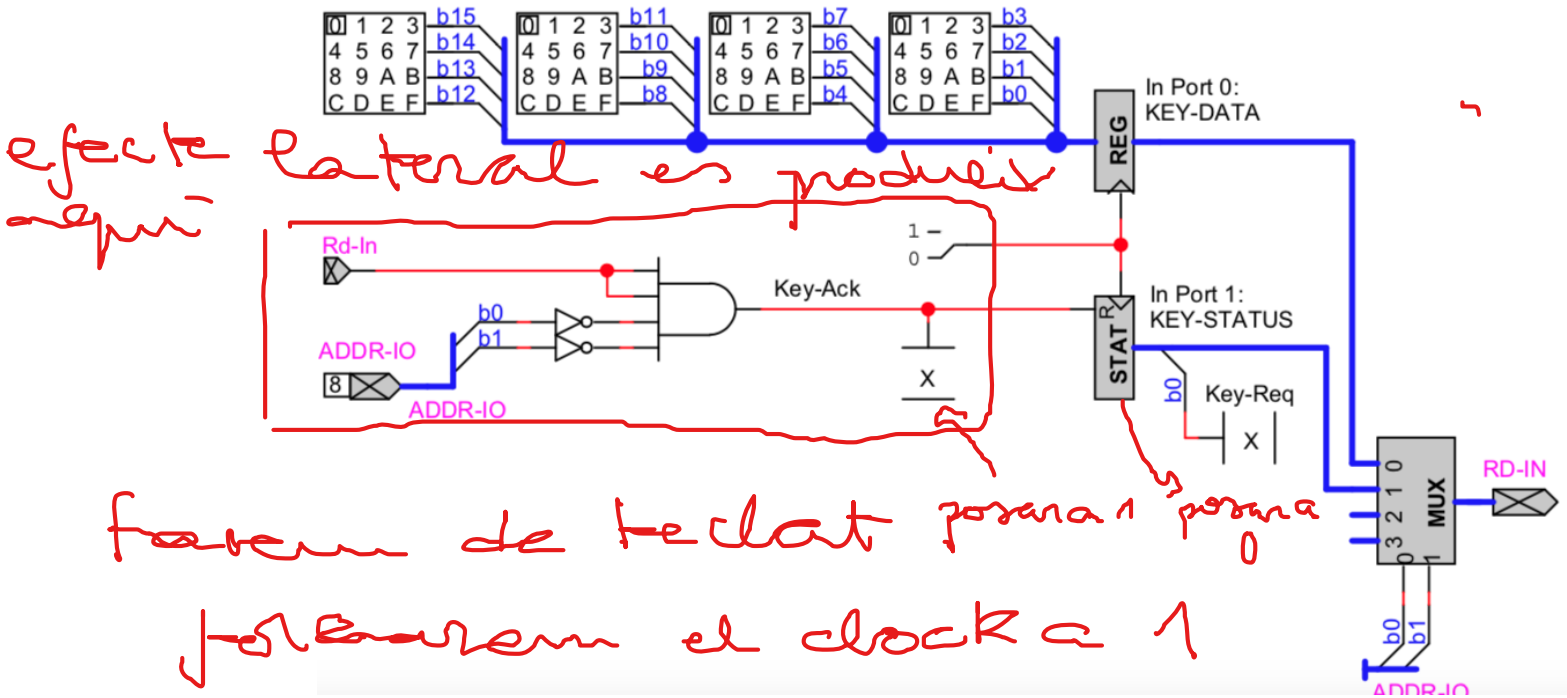
Registres biestables no tenen senyal asincrona

tenim excepcions registres amb entrada asincrona set i reset. Mai poden ser =



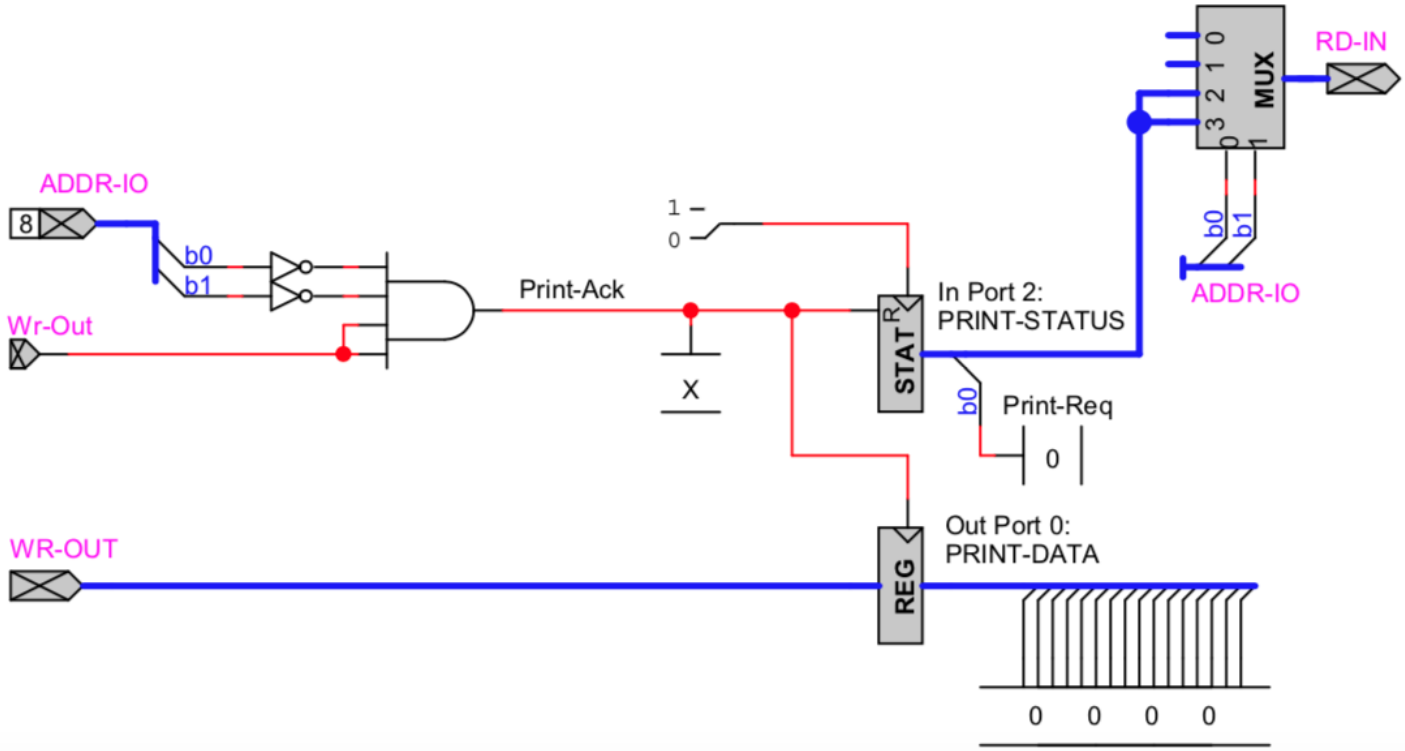
Efecte lateral llegint *port* dades teclat

- Key-Req es posa a "1" amb el *Binary switch* que fa de Clk als *ports*
- Al fer `IN` sobre el *port* KEY-DATA
  - El valor de KEY-DATA surt per RD-IN
  - La AND-4 fa que Key-Ack valgui "1"
    - Té l'efecte lateral que Key-Req, *request* del teclat, torni a "0"



Efecte lateral escrivint *port* dades impressor

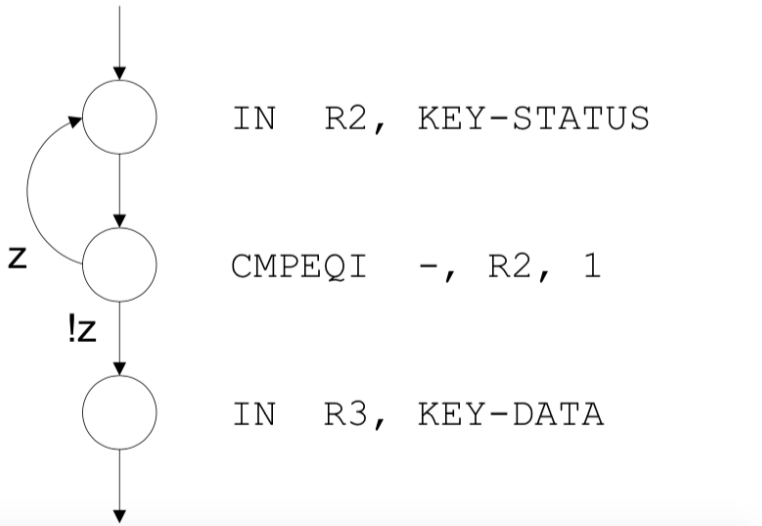
- Print-Req es posa a "1" amb el *Binary switch* que fa de Clk als *ports*
- Al fer `OUT` sobre el *port* de PRINT-DATA
  - S'escriu el valor del bus WR-OUT al *port* PRINT-DATA
  - La AND-4 fa que Print-Ack val "1"
    - Té l'efecte lateral que Print-Req, *request* de la impressora, torni a "0"



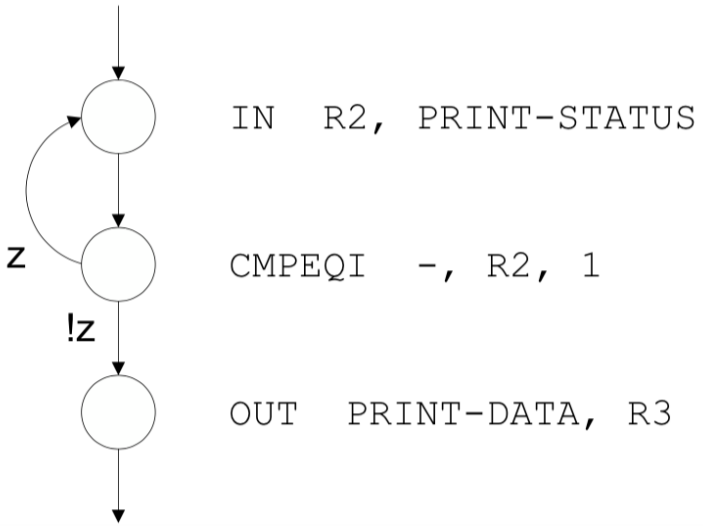
Efecte lateral: accions i graf d'estats UC

- Fragments graf d'estats UC
  - KEY-DATA i PRINT-DATA són constants que representen els identificadors dels *ports* de dades de teclat i impressora
  - KEY-STATUS i PRINT-STATUS són constants que representen els identificadors dels *ports* d'estat del teclat i impressora, senyal *request* és al bit de menys pes del *port* (i resta de bits del *port* a "0")

Lectura de teclat (a R3)



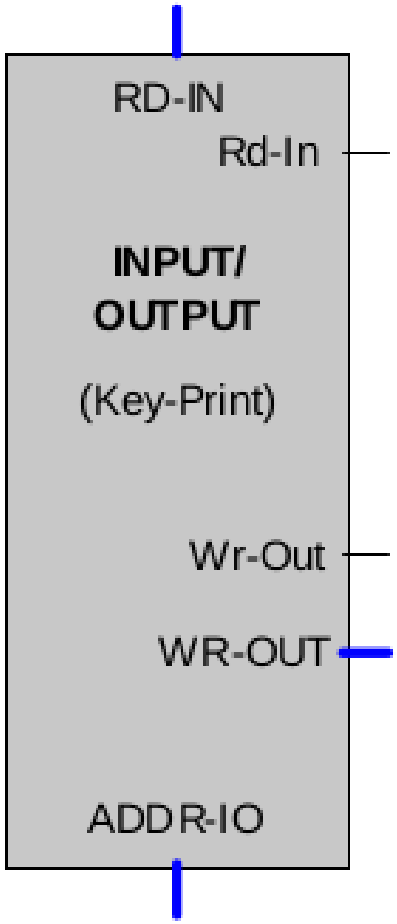
Escriptura a impressora (de R3)



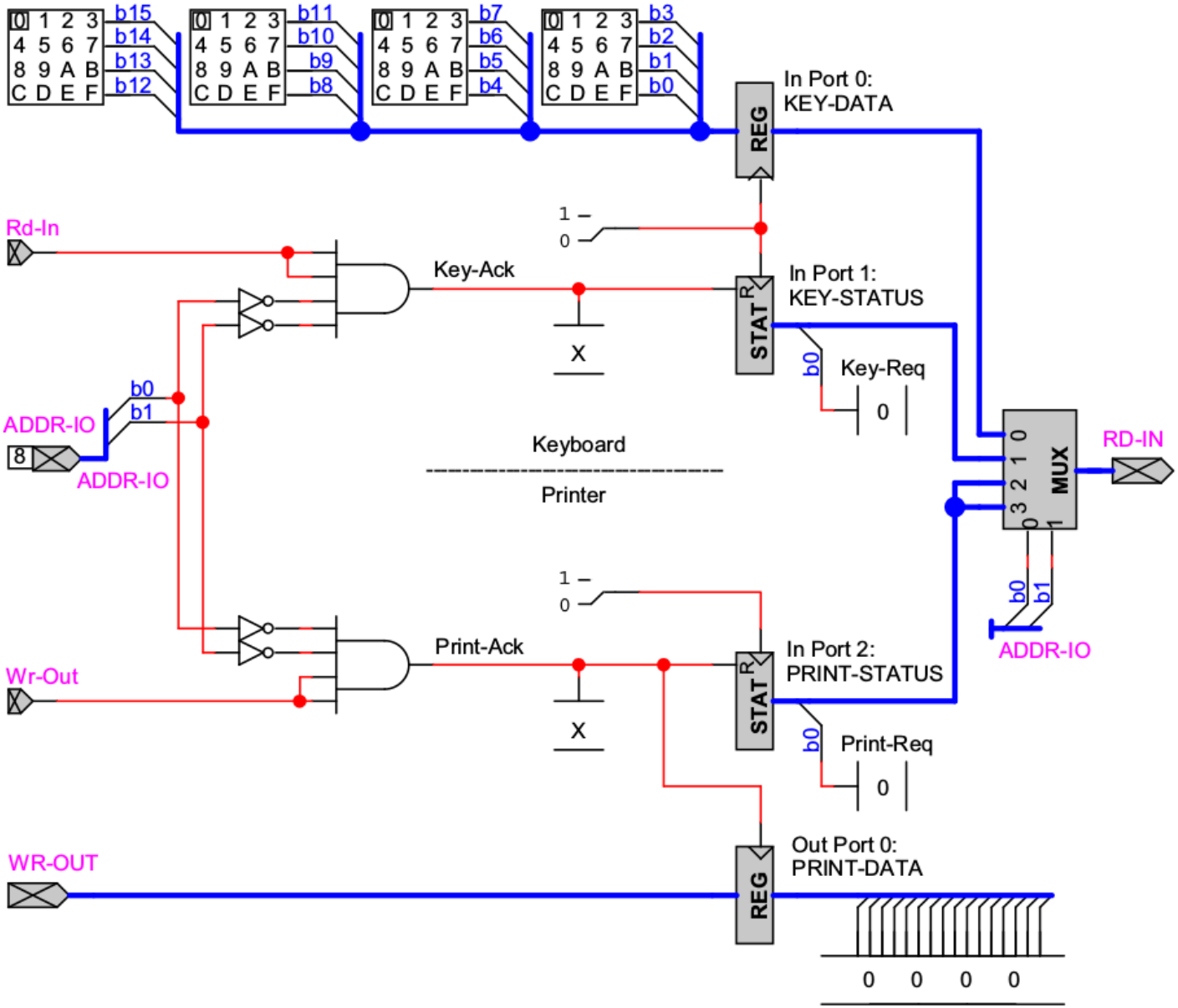
Índex

Subsistema E/S: encapsulat

- Senyals d'entrada:
  - Senyals de control Rd-In i Wr-Out
  - Bus d'adreça ADDR-IO
  - Bus de dades WR-OUT
- Senyals de sortida:
  - Bus de dades RD-IN



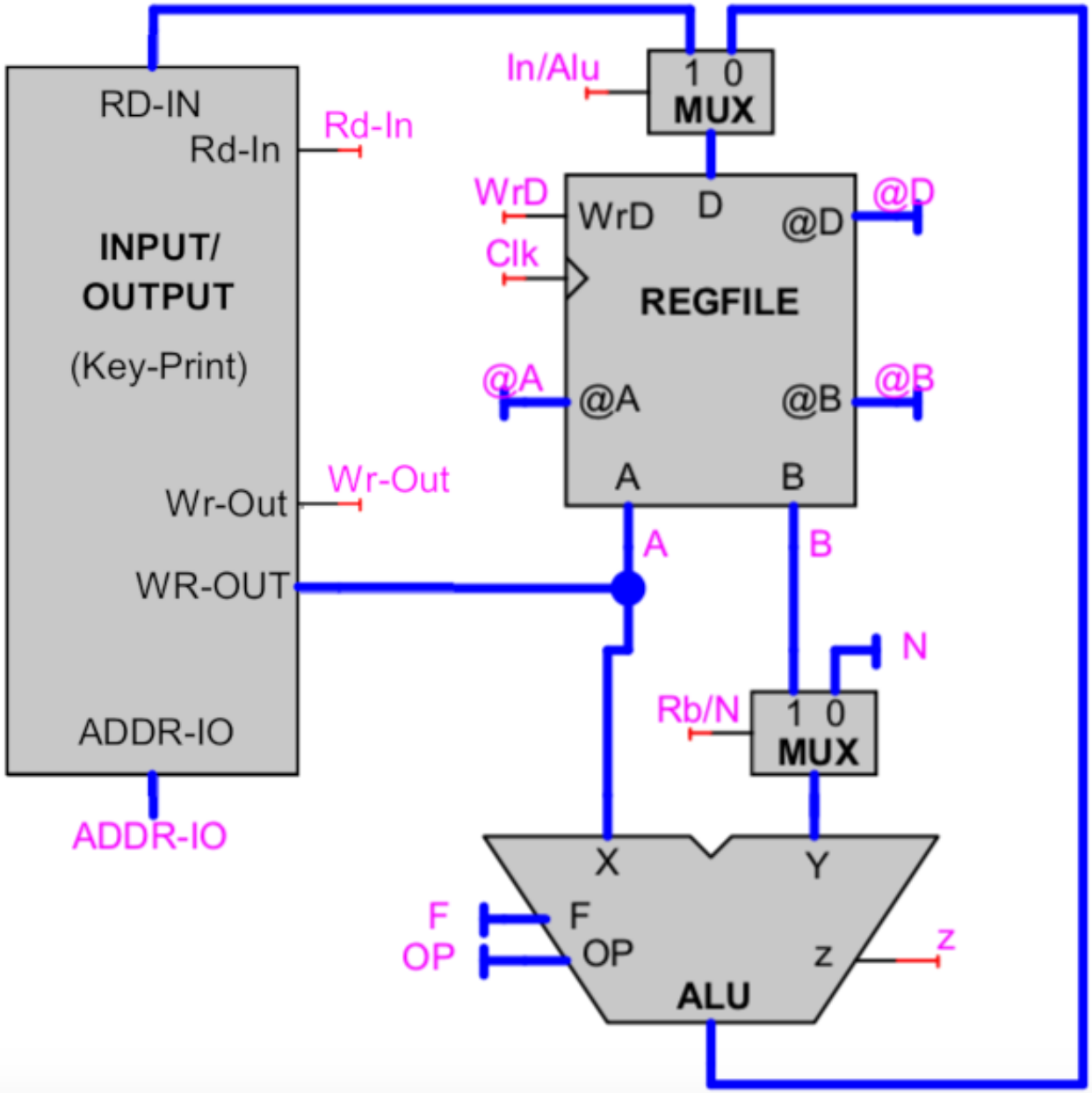
Subsistema E/S: esquema final



Subsistema E/S: observacions

- Identificadors de *ports*:
  - D'entrada:
    - 0 (KEY-DATA): *port* de dades de teclat
    - 1 (KEY-STATUS): *port* d'estat del teclat
    - 2 (PRINT-STATUS): *port* d'estat de la impressora
  - De sortida:
    - 0 (PRINT-DATA): *port* de dades de la impressora
- Com utilitzem pocs ports, la implementació del mòdul INPUT/OUTPUT descarta els 6 bits alts i només considera els 2 bits baixos de ADDR-IO

Esquema UPG + I/O Key-Print



Demostració: E/S a Logic-Works

- Descripció del programa:
  - Llegeix un nombre de teclat `N`
    - Hem de generar manualment el *request* amb el *Binary Switch* per indicar que hi ha una dada disponible al teclat
  - A continuació llegeix `N` bytes del teclat
    - Hem de generar manualment els *request* amb el *Binary Switch* després de cada *byte* per indicar que hi ha una dada disponible al teclat
    - Els emmagatzema a memòria (s'explicarà als propers temes)
  - Finalment, mostra els `N` bytes per impressora
    - Hem de generar manualment els *request* amb el *Binary Switch* després de cada *byte* per indicar que la impressora està disponible
- Video captura:
  - `http://personals.ac.upc.edu/enricm/Docencia/IC/IC9_demo.mp4`
  - $N = 4$, $dades = 0x40, 0x41, 0x42, 0x43$
  - A 0:19 i 1:02 es pot apreciar l'efecte lateral a la lectura i escriptura

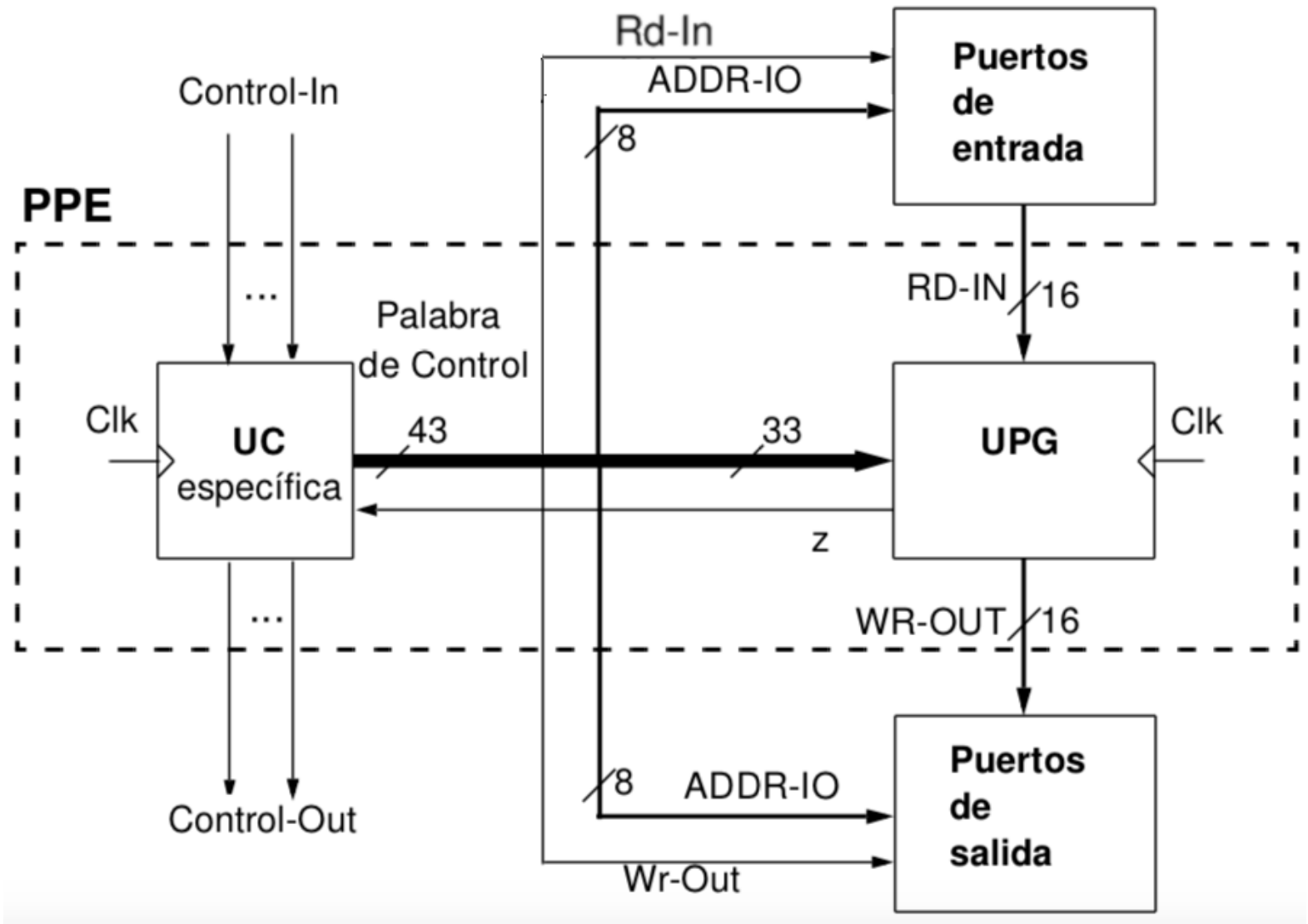
Nova paraula de control

- S'afegeixen 10 bits a la paraula de control del tema anterior
  - ADDR-IO: bus de 8 bits amb l'identificador de *port*
  - Wr-Out: senyal binari que indica si en aquest cicle es fa l'acció `OUT`
  - Rd-In: senyal binari que indica si en aquest cicle es fa l'acció `IN`
- La nova paraula de control té 43 bits:

	@A			@B			Rb/N	OP		F			In/Alu	@D			WrD	Wr-Out	Rd-In	N (hexa)				ADDR-IO (hexa)	
`IN R2, 33`	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	1	0	1	X	X	X	X	2	1
`OUT 0x50, R1`	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	X	X	X	X	5	0
`ADD R1, R2, R3`	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	1	1	0	0	X	X	X	X	X	X

wrd →
wr_out_mai X
Rd_In →

Connexió UC / UPG

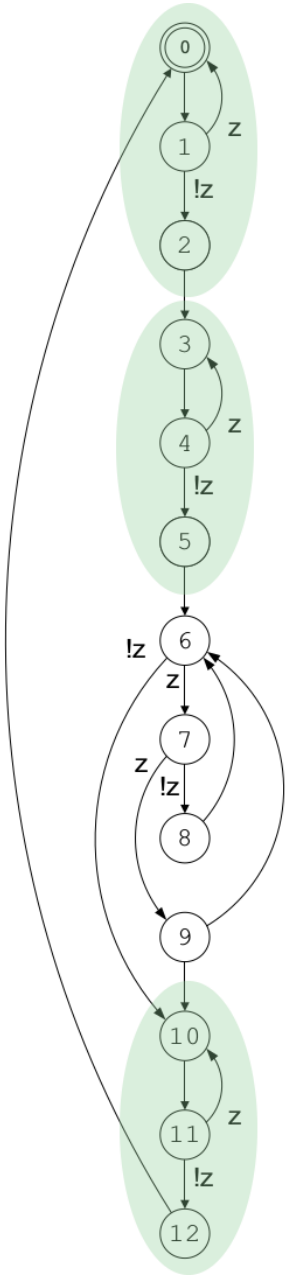


Índex

MCD: especificació

- Fer el graf d'estats de la UC que calcula el MCD
- Especificació:
  - Els dos operands es llegiran del teclat
    - Caldrà fer una espera activa per obtenir cada operand
  - El resultat es mostrarà per la impressora
    - Caldrà fer una espera activa per esperar que estigui disponible

MCD: Graf d'estats UC amb E/S




```
IN      R0, KEY-STATUS
CMPEQI  -, R0, 1
IN      R0, KEY-DATA
IN      R1, KEY-STATUS
CMPEQI  -, R1, 1
IN      R1, KEY-DATA
CMPEQ   -, R0, R1
CMPLTU  -, R1, R0
SUB     R0, R0, R1
SUB     R1, R1, R0
IN      R2, PRINT-STATUS
CMPEQI  -, R2, 1
OUT     PRINT-DATA, R0
```

Índex

Exercicis típics

- Conversió entre paraula de control de 43 bits i mnemotècnic
  - Paraula de control corresponent a
- Afegir E/S asíncrona a PPE
  - Fer l'entrada/sortida utilitzant teclat/impressora

Exercicis a entregar a Atenea

- Enunciat disponible a Atenea
  - `https://atenea.upc.edu/pluginfile.php/3603444/mod_assign/introattachment/0/Tema%209%20-%20Exercicis%20en%20paper.pdf?forcedownload=1`
- Entrega a Atenea fins el dimecres 18/11
  - 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/`
  - Els esquemes lògics els podeu fer a mà i posteriorment fotografiar-los/escanejar-los o utilitzar alguna eina d'edició de circuits (Logic Works, ...)

Índex

Conclusions

- Hem afegit al nostre computador subsistema d'E/S
  - Des del punt de vista de la UPG, un perifèric serà un conjunt de *ports*
    - D'entrada (de dades i d'estat), de sortida (de dades)
    - Identificats amb un nombre natural de 8 bits, ADDR-IO
- La comunicació entre UPG i perifèrics serà asíncrona
  - *Four-phase handshaking*
  - Utilitza senyals de control *request* i *acknowledgment*
  - Els senyals *request* s'emmagatzemaran als *ports* d'estat
- La UC generarà una paraula de control de 43 bits
  - Afegim camps ADDR-IO (8b), Wr-Out (1b) i Rd-In (1b)
- **Gràcies a l'efecte lateral a la lectura/escriptura als *ports* de dades, cada comunicació només requereix de tres nodes al graf d'estats de la UC**
  - **Dos nodes per fer una espera activa sobre el *port* d'estat**
  - **Un node per fer la transferència de dades**
- No oblideu fer el qüestionari ET9 i els exercicis en paper (slide 45)

necessitem 3 estats per entrada i 3 estats per sortida

Índex

- Sincronització per enquesta vs. sincronització per interrupcions
  - Hem vist la sincronització per enquesta
  - Un altre tipus de sincronització és per interrupcions
    - En general, més eficient que per enquesta
    - La veureu a EC
  - Analogia:
    - Enquesta ≡ Mirar si tenim correu postal a la bústia
    - Interrupcions ≡ Rebre una trucada telefònica
- Protocol *Two-phase handshake*
  - Variant del protocol *four-phase handshake*
  - Quan *request* i *acknowledgment* passen de "1" a "0" es fa una nova transferència
    - *No Return to zero* (NRZ)
    - A IC no l'utilitzarem

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://www.algru.es/default/buzones/buzones-verticales/buzones-verticales-interior.html.

[2] [Online]. Available: https://www.amazon.es/GPO-746-Rotary-Telephone-Importado/dp/B00YAE5K48/ref=asc_df_B00YAE5K48.

Introducció als Computadors

Tema 9: Entrada/Sortida

http://personals.ac.upc.edu/enricm/Docencia/IC/IC9.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.

Presentació publicada sota llicència Creative Commons 4.0