Introducció als Computadors

Tema 8: Unitat de Procés General (UPG)

http://personals.ac.upc.edu/enricm/Docencia/IC/IC8b.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: UPG com a eina multiusos

[1]

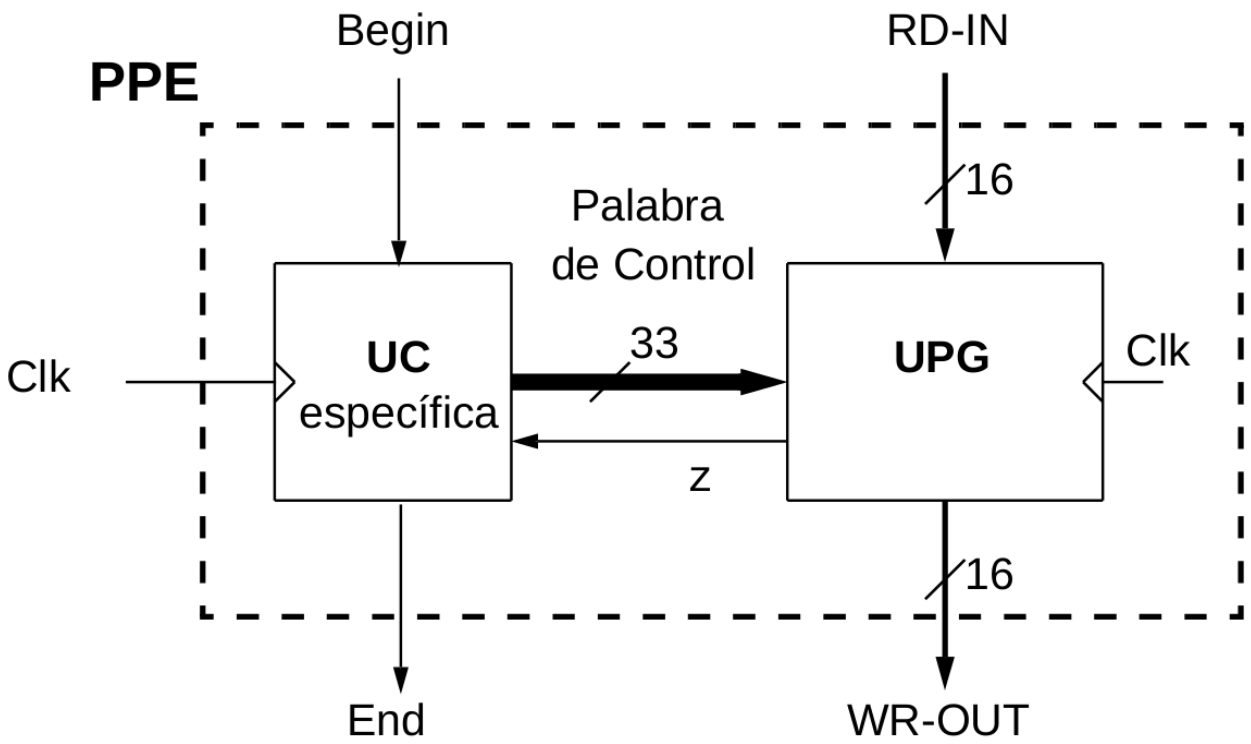
Índex

Introducció

- Hem iniciat el disseny d'una Unitat de Procés General (UPG)
  - Té una paraula de control de 33 bits i genera 1 bit de condició (z)
  - La UC haurà de generar cada cicle la paraula de control
  - El bit z permet decidir quin és l'estat següent de la UC
- Ja hem vist com passar del codi C al graf d'estats de la UC
- Ara incorporarem l'E/S síncrona del PPE al graf d'estats de la UC
  - El mateix senyal de rellotge sincronitza emissor i receptor
    - Al tema 9 parlarem d'E/S asíncrona
- Adaptarem els PPE's del tema anterior a la UPG amb E/S
  - Crearem el graf d'estats de la UC
  - Potser haurem de modificar el sincronisme d'E/S
    - Per exemple, MCD tenia dos bussos d'entrada de dades

Interfície dels PPE's que utilitzen la UPG

- El PPE té un senyal de control d'entrada (*Begin*) i un de sortida (*End*)



Graf d'estats de la UC amb E/S

- Crearem el graf d'estats de la UC
  - Cada estat tindrà dos senyals binàris d'entrada
    - El senyal de control d'entrada *Begin*
    - El bit de condició *z*
  - Cada estat generarà 34 senyals binaris de sortida
    - La paraula de control (33 bits) / Mnemotècnic
    - El senyal de control de sortida *End* (1 bit)
- La llegenda de tots els grafs d'estats d'aquest tema serà:

  - Si a un arc no apareix el nom d'un senyal d'entrada, indica que la transició és independent del valor d'aquest senyal
  - Si el nom apareix sense negar, la transició es realitza si el senyal val "1"
  - Si el nom apareix negat, la transició es realitza si el senyal val "0"

Índex

Índex

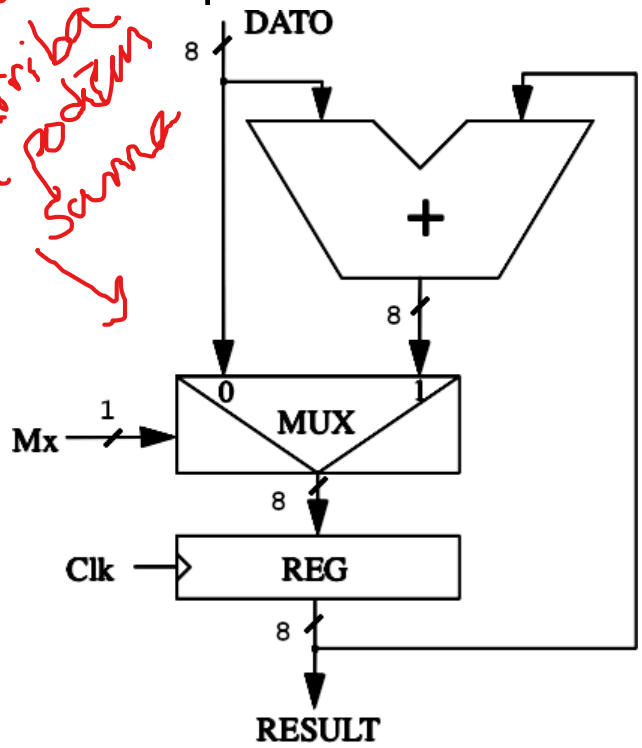
Suma-4: especificació

- La mateixa que al tema 7:
  - El cicle en que *Begin* val "1", a RD-IN també tindrem el primer valor
  - En cicles consecutius es rebran el segon, el tercer i el quart valor
  - El cicle que es mostri el resultat per WR-OUT, *End* ha de valdre "1"
    - La resta de cicles *End* ha de valdre "0"
  - El mateix cicle que es mostra el resultat d'una Suma-4, es pot començar a calcular una nova Suma-4
  - S'ignorarà el valor de *Begin* mentre es realitza una Suma-4
- L'especificació no determina quant de temps triga en donar el resultat
  - Qui utilitzi aquest PPE ha d'esperar fins que *End* valgui "1"

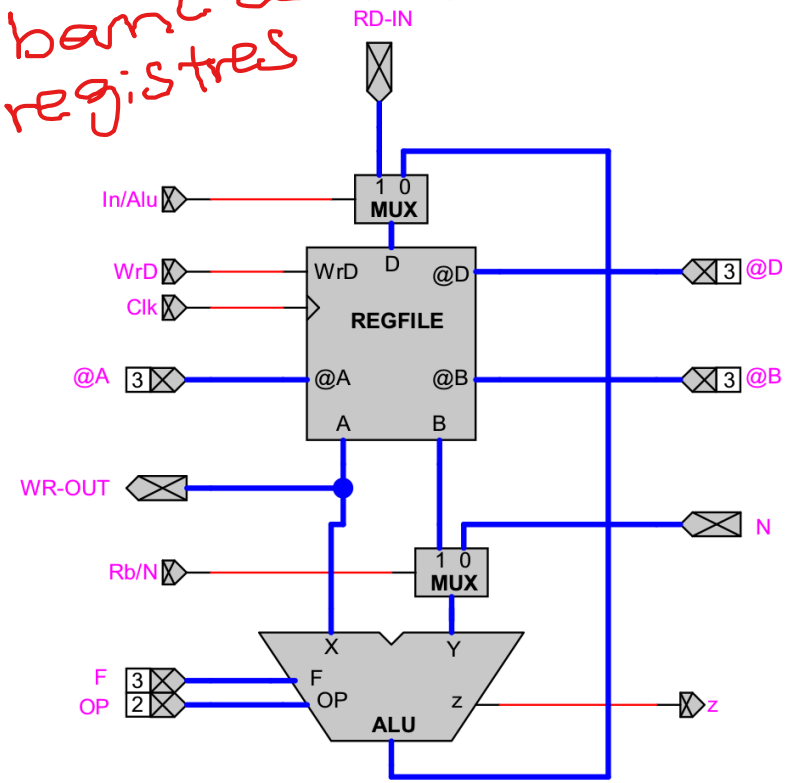
Cicle	0	1	2	3	4		t	t+1
Begin	0	1	x	x	x	...	0	0
RD-IN (dec)	x	23	5	12	18	...	x	x
WR-OUT (dec)	x	x	x	x	x	...	58	x
End	0	0	0	0	0	...	1	0

Suma-4: UP específica vs. UPG

UP específica Suma-4



UPG

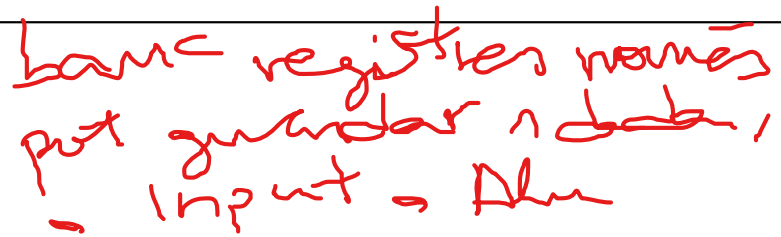


- A la UP específica, el mateix cicle que rebem una dada de l'exterior podem operar amb ella
  - A la UPG, abans de poder operar amb ella ha de passar pel REGFILE
  - A més, al REGFILE només puc fer una escriptura per cicle

Suma-4: implementació a la UPG

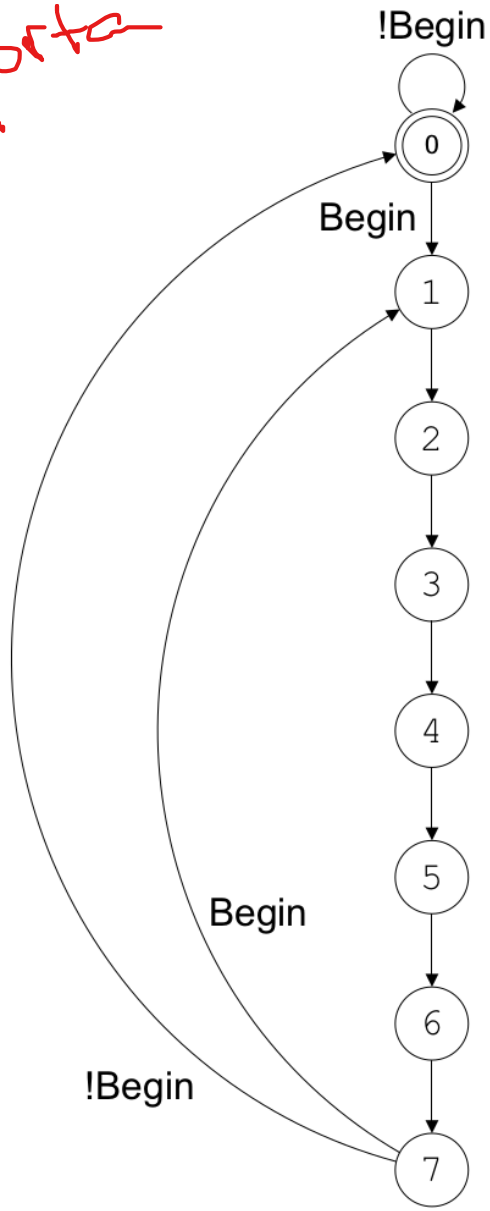
- Com la UPG té 8 registres, podem guardar els 4 valors a 4 registres
- **No** podrem fer sumes parcials a mesura que rebem els valors perquè no podríem guardar el resultat al banc de registres
  - Ja estem escrivint al REGFILE la dada rebuda pel bus RD-IN
- Un cop rebuts els 4 nombres, podrem sumar-los dos a dos
  - Fent una suma per cicle
- Assumim que al cicle t, *Begin*="1"

```
R1 = RD-IN(t);
R2 = RD-IN(t+1);
R3 = RD-IN(t+2);
R4 = RD-IN(t+3);
R5 = R1 + R2;          // t+4
R5 = R3 + R5;          // t+5
R5 = R4 + R5;          // t+6
WR-OUT(t+7) = R5;
```



- Calen tres cicles per sumar els quatre valors
- A l'instant t+7, mostra el resultat per WR-OUT i posa *End* a "1"

Suma-4: graf d'estats UC per a la UPG




```
IN   R1              // End = 0
IN   R2              // End = 0
IN   R3              // End = 0
IN   R4              // End = 0
ADD R5, R1, R2       // End = 0
ADD R5, R3, R5       // End = 0
ADD R5, R4, R5       // End = 0
OUT R5 // IN R1 // End = 1
```

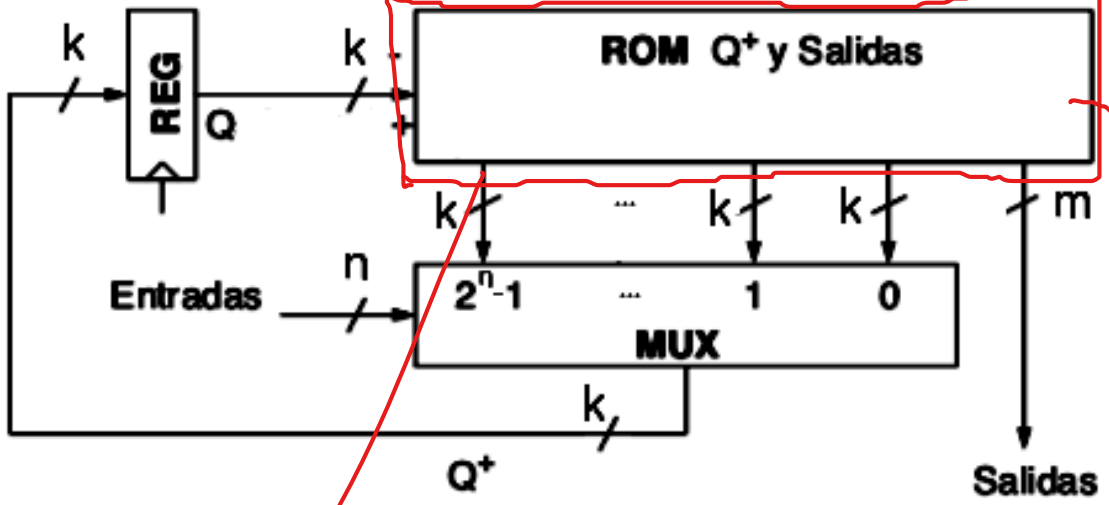
Suma-4: paraules de control

Mnemotécnicos	Palabra de Control de 33 bits																																	
	@A			@B			Rb/N	OP		F			In/Alu	@D			WrD	N (hexa)			
IN R1	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	X	X	X	X
IN R2	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	1	X	X	X	X
IN R3	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	1	X	X	X	X
IN R4	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	1	X	X	X	X
ADD R5, R1, R2	0	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	1	X	X	X	X
ADD R5, R3, R5	0	1	1	1	0	1	1	0	0	1	0	0	0	1	0	1	1	X	X	X	X
ADD R5, R4, R5	1	0	0	1	0	1	1	0	0	1	0	0	0	1	0	1	1	X	X	X	X
OUT R5 // IN R1	1	0	1	x	x	x	x	x	x	x	x	x	1	0	0	1	1	X	X	X	X

out i in a la vegada

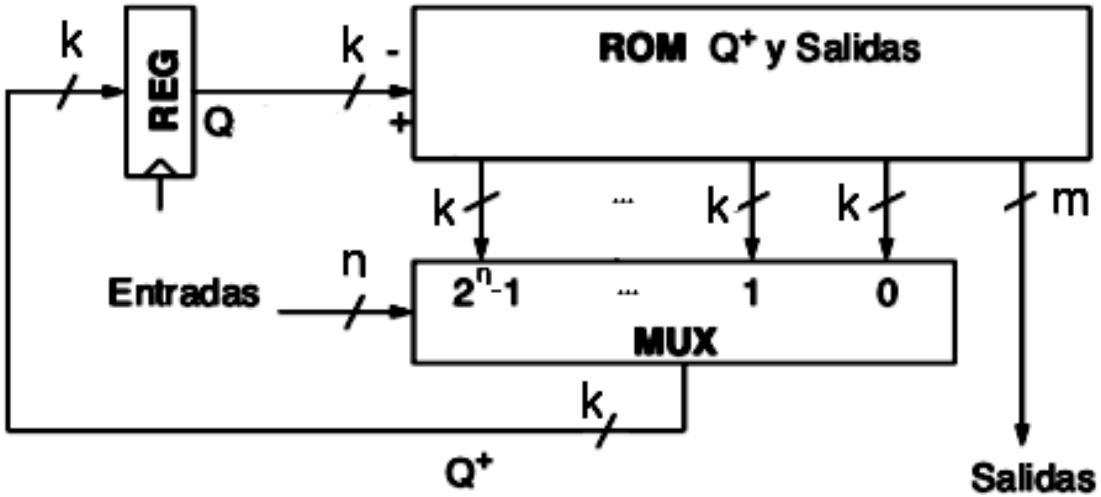
Suma-4: mida síntesi UC

- Sintetitzem la UC amb una única ROM i un multiplexor de bussos. Quina serà la seva mida mínima?



Suma-4: mida síntesi UC

- Sintetitzem la UC amb una única ROM i un multiplexor de bussos. Quina serà la seva mida mínima?



- Solució:
  - Parametrització dels busos:
    - bits d'entrada: $n = 2$ (*Begin* i el bit de condició z)
    - bits de sortida: $m = 34$ (els 33 de la paraula de control i *End*)
    - bits d'estat: $k = \lceil log_2 8 \rceil = 3$
  - La ROM tindrà $2^k = 8$ paraules (files)
  - Cada paraula tindrà $2^n \times k + m$ bits, és a dir $2^2 \times 3 + 34 = 46$ bits
  - La mida de la ROM serà 8 *paraules* $\times$ 46 *bits*/*paraula* = 368 bits

Suma-4: observacions

- El PPE Suma-4 del tema 7 triga 4 cicles però el del tema 8 en triga 7
  - La UP específica opera directament amb la dada del bus de dades
  - A la UPG, la dada ha de passar prèviament pel banc de registres
    - El banc de registres només permet una escriptura per cicle
  - Si el disseny de la UPG hagués fet arribar RD-IN a l'entrada de la ALU, també podria fer el càlcul en 4 cicles
- Qualsevol UC de fins a 8 estats que controli la UPG es pot representar amb 368 bits
  - El contingut de la "ROM Q^+ i sortides"
  - Una matriu de bits de 8 files $\times$ 46 columnes = 368 bits
  - Aquests 368 bits constitueixen el *programa* que controla la UPG

Índex

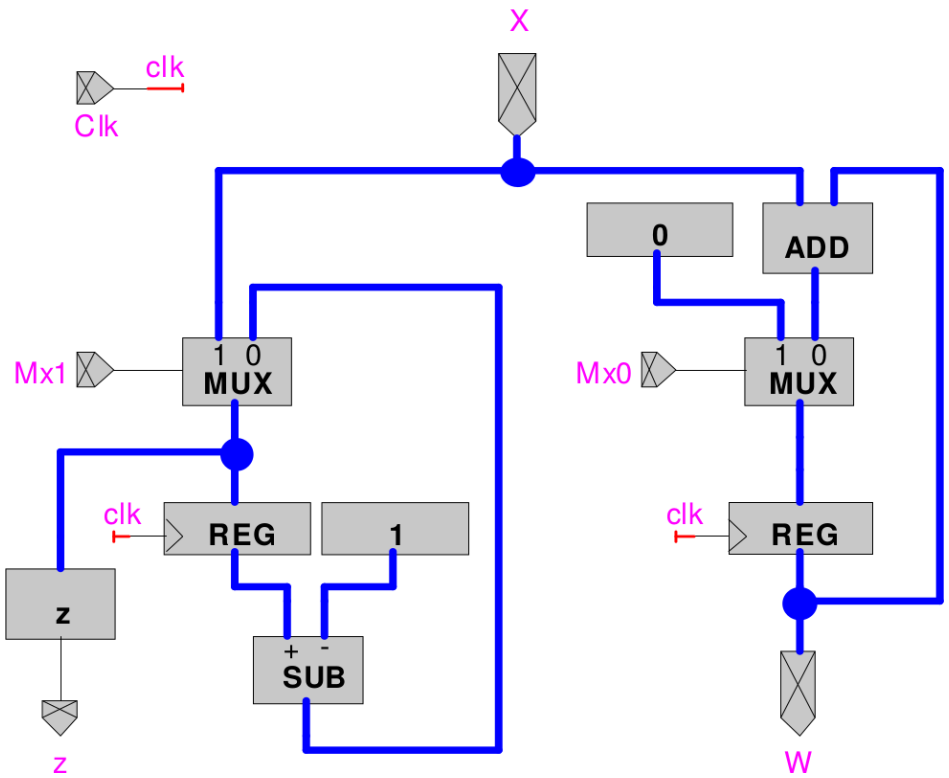
Suma-N: Especificació

- El mateix cicle que *Begin* val "1", a RD-IN es rep el valor N
- Els següents N cicles, es rebran els N valors a sumar
- El cicle que es mostri el resultat per WR-OUT, *End* ha de valdre "1"
  - La resta de cicles *End* ha de valdre "0"
- El mateix cicle que es mostra el resultat d'una Suma-N, es pot començar a tractar una nova Suma-N
- S'ignorarà el valor de *Begin* mentre es realitza un Suma-N

Cicle	0	1	2	3	4	5	6		t	t+1
Begin	0	1	x	x	x	x	x	...	0	0
RD-IN (dec)	x	5	15	12	18	10	22	...	x	x
WR-OUT (dec)	x	x	x	x	x	x	x	...	77	x
End	0	0	0	0	0	0	0	...	1	0

Suma-N: PPE amb UP específica

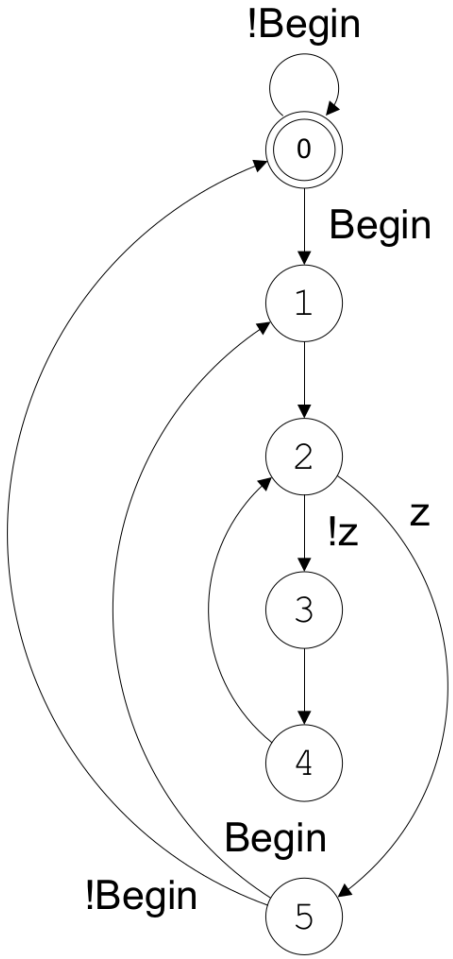
- Implementació:
  - Dos registres
    - Un conté la suma parcial i l'altre el comptador de nombres pendents
  - Dos blocs aritmètics
    - Un bloc actualitza la suma i l'altre el comptador
    - Els blocs treballen en paral·lel



Suma-*N*: implementació a la UPG

- **No** podrà complir l'especificació original
  - Com hem vist al Suma-4, les dades rebudes per RD-IN s'han de guardar a REGFILE al mateix cicle
  - Però la UPG només té 8 registres!
  - A més, la UPG només pot fer un càlcul per cicle
- Caldrà relaxar la temporització
  - Abans de rebre un nou nombre, caldrà haver processat l'anterior
    - Carregar el nombre en un registre
    - Actualitzar la suma parcial
    - Actualitzar el comptador de nombres pendents
    - Decidir si es continua iterant o si es pot abandonar el bucle

Suma-N: graf d'estats UC per a la UPG




```
IN   R1           // End = 0
IN   R2           // End = 0
ADDI R1, R1, -1   // End = 0
IN   R3           // End = 0
ADD  R2, R2, R3   // End = 0
OUT  R2 // IN R1 // End = 1
```

Suma-N: observacions

- Paper dels registres:
  - R1 s'inicialitza amb el valor N i després conté la quantitat de nombres pendents de ser tractats
  - R2 s'inicialitza amb el primer nombre i després conté la suma parcial dels nombres llegits fins ara
  - R3 s'utilitza per llegir la resta de nombres
- Temporització aconseguida:
  - Suposant que en el cicle c Begin="1" i es rep el valor N,
    - al cicle $c + 1$ es rep el primer nombre
    - al cicle $c + 3$ es rep el segon
    - a partir d'ara, cada tres cicles es rep un nou nombre
- L'acabament del bucle es detecta quan el comptador arriba al valor 0
  - El bit z ho indica a l'estat on es decrementa el comptador
- Si el codi incrementés el comptador en comptes de decrementar-lo...
  - El final del bucle es detectaria quan el comptador arribés a N
    - Caldria un estat més per fer la comparació del comptador amb N
    - La UPG processaria els nombres a un ritme d'un cada quatre cicles

Suma-N: exercicis

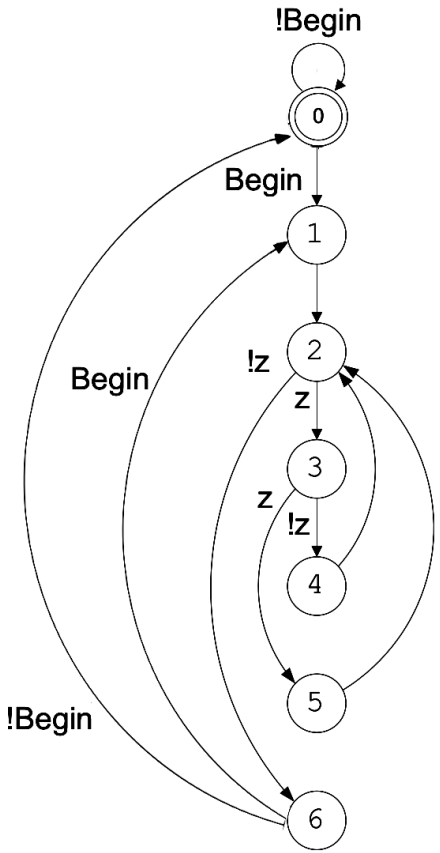
- Determineu les paraules de control corresponents als mnemotècnics del graf d'estats de la UC que calcula Suma-N
- Quants cicles triga el càlcul si $N = 5$?
  - Des del cicle on $Begin = 1$ fins el cicle on $End = 1$, ambdós inclosos
- Quants cicles triga el càlcul per a un valor arbitrari de N, on $N \geq 2$?
- Com caldria modificar el graf d'estats de la UC si, suposant que en el cicle c Begin="1" i es rep el valor N, al cicle $c + 3$ es rep el primer nombre, al cicle $c + 6$ es rep el segon, al cicle $c + 9$ el tercer, ...?

Índex

MCDv1: especificació

- Haurem d'adaptar el protocol d'entrada sortida del PPE
  - El PPE del tema 7 tenia dos bussos d'entrada de dades però la UPG només en té un (RD-IN)
- Quan *Begin* valgui "1", aquest cicle rebrem el primer nombre a RD-IN
  - El segon nombre es rebrà el cicle següent
- Quan el càlcul estigui complet, *End* es posarà a "1" i, al mateix cicle, també mostrarà el resultat a WR-OUT

MCDv1: graf d'estats UC per a la UPG




```
IN      R0                  // End = 0

IN      R1                  // End = 0

CMPEQ   -, R0, R1           // End = 0

CMPLTU  -, R1, R0           // End = 0

SUB     R0, R0, R1          // End = 0

SUB     R1, R1, R0          // End = 0

OUT      R0   // IN      R0 // End = 1
```

MCDv1: observacions

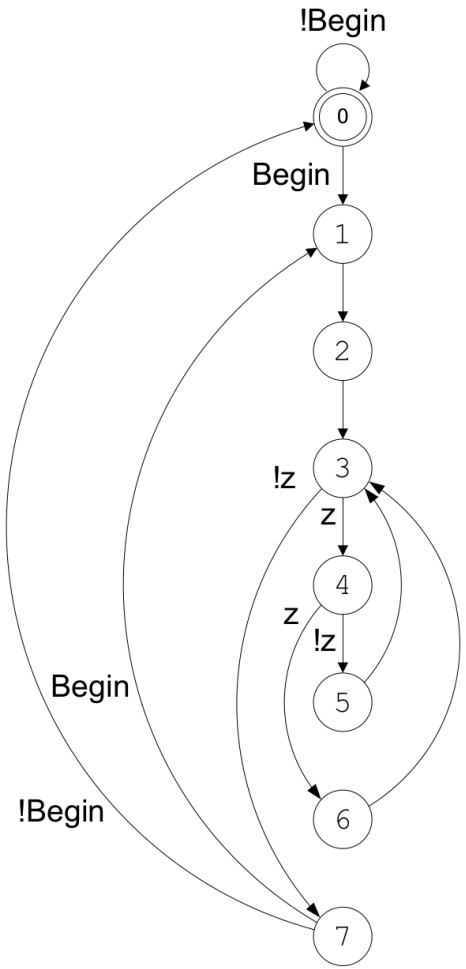
- A l'estat 0 es carrega RD-IN a R0 per si al final de cicle *Begin*="1"
  - Si no ho féssim, perdríem el primer operand del MCD
- A l'estat 1 es carrega el segon operand a R1
- Els estats 2 al 5 els vàrem veure com a exemple a l'anterior tema
- A l'estat 6:
  - es mostra el resultat per WR-OUT
  - es posa a "1" el senyal del control *End*
  - es carrega RD-IN a R0 per si al final d'aquest cicle *Begin*="1"
- L'estat següent a l'estat 6 depèn del valor del senyal *Begin*
  - Si *Begin*="1" l'estat següent és 1 perquè s'ha iniciat un nou càlcul
  - Si *Begin*="0" l'estat següent és 0 per esperar l'inici d'un nou càlcul

MCDv2: especificació

- Modificarem el protocol d'E/S del MCDv1
  - *Begin* i *End* estaran un cicle avançades a les dades
- Quan *Begin* valgui "1" indicarà que **al cicle següent** rebrem el primer nombre a RD-IN
  - El segon nombre es rebrà dos cicles després
- Quan el càlcul estigui complet *End* es posarà a "1" i, **al cicle següent** mostrarà el resultat a WR-OUT

MCDv2: graf d'estats UC per a la UPG

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona




```
OUT     R0                  // End = 0

IN      R0 // OUT     R0    // End = 0

IN      R1                  // End = 0

CMPEQ   -, R0, R1           // End = 0

CMPLTU  -, R1, R0           // End = 0

SUB     R0, R0, R1          // End = 0

SUB     R1, R1, R0          // End = 0

NOP                         // End = 1
```

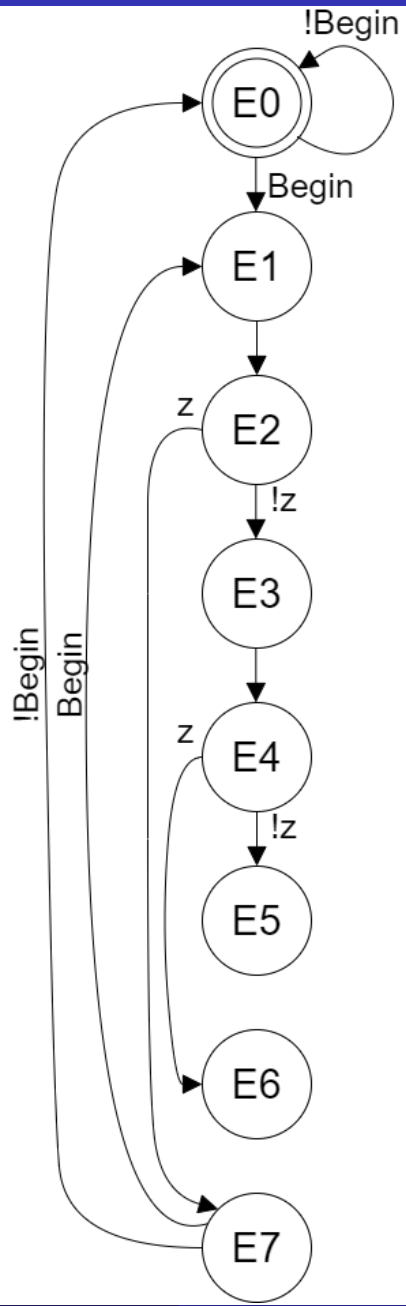
MCDv2: observacions

- A aquesta temporització, l'estat 0 no carrega RD-IN a cap registre
  - Els estats 1 i 2, carreguen els operands a R0 i R1
- Quan des de l'estat 3 es detecta que el càlcul ha finalitzat, salta a l'estat 7, on s'activa el senyal *End*
  - Com a aquest estat no es pot fer res més, la paraula de control serà `NOP`
  - A l'estat següent a l'estat 7 es mostrarà el resultat per WR-OUT
    - Però com podem encadenar un nou càlcul, l'estat següent pot ser 0 o 1
    - Per tant, a tots dos estats s'haurà de fer `OUT`
    - No pot portar a confusió a l'exterior del PPE perquè únicament un dels dos `OUT`'s anirà precedit per *End*="1"
- Les dues accions de l'estat 1 es poden fer en paral·lel

Màxim-N: especificació

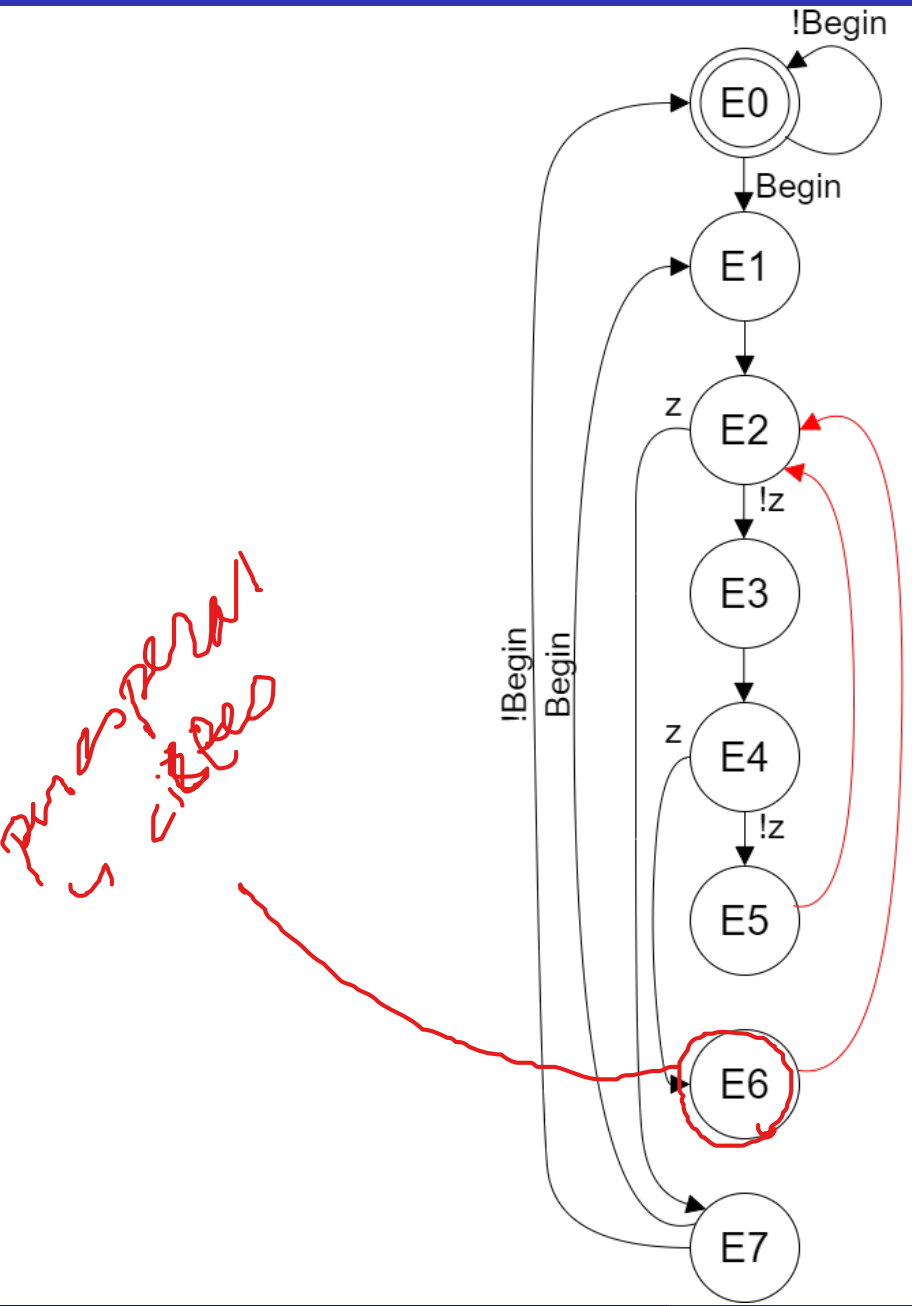
- Completeu el disseny de la unitat de control específica d'un PPE que permeti rebre N enters i calcular-ne el màxim.
- El primer valor que arriba és N (on $N \geq 1$) i estarà disponible en RD-IN al mateix cicle que el senyal *Begin* valgui "1" (cicle k).
- El primer nombre pel càlcul arriba al cicle següent (cicle k+1).
- Si n>1, el segon nombre arriba al cicle k+3 i la resta de nombres arriben cada 4 cicles (és a dir, als cicles k+7, k+11, k+15, etc.).
- Una vegada s'ha iniciat un càlcul, s'ha d'ignorar el senyal *Begin* fins que aquest acabi.
- Al mateix cicle en què el resultat està disponible a la sortida WR-OUT, i per tant el senyal *End* es posa a "1"
  - Si al final d'aquest cicle *Begin* val "1", es començarà un nou càlcul.

Completeu el graf d'estats de la UC




```
IN R0              // End = 0

.............................

.............................

IN R2              // End = 0

CMPLT -, R1, R2 // End = 0

.............................

NOP                // End = 0

OUT R1 // IN R0 // End = 1
```

Completeu el graf d'estats de la UC




```
IN R0                // End = 0
IN R1                // End = 0
SUBI R0, R0, 1       // End = 0
IN R2                // End = 0
CMPLT -, R1, R2      // End = 0
MOVE R1, R2          // End = 0
NOP                  // End = 0
OUT R1 // IN R0 // End = 1
```

Màxim-N: observacions

- Paper dels registres:
  - R0 s'inicialitza amb el valor N i després conté la quantitat de nombres pendents de ser tractats
  - R1 s'inicialitza amb el primer nombre i després conté el valor màxim detectat fins ara
  - R2 s'utilitza per llegir la resta de nombres
- A E2 es decrementa R0 i amb el bit z es decideix la sortida del bucle
- E3 i E4 obtenen un nou nombre i el comparen amb el màxim actual
- Si el nou nombre és el nou màxim, E5 el guarda a `R1` i es torna a iterar
- L'estat E6 amb l'acció `NOP` garanteix que totes les iteracions del bucle triguen exactament 4 cicles
  - Tipus d'iteracions possibles:
    - Si el nombre llegit és nou màxim: $E3 \rightarrow E4 \rightarrow E5 \rightarrow E2$
    - Si el nombre llegit no és nou màxim: $E3 \rightarrow E4 \rightarrow E6 \rightarrow E2$
  - Garanteix sincronisme del PPE Màxim-N amb les dades que arriben de l'exterior a raó d'una cada quatre cicles

Índex

No farem mai entrades síncrones

És inviable

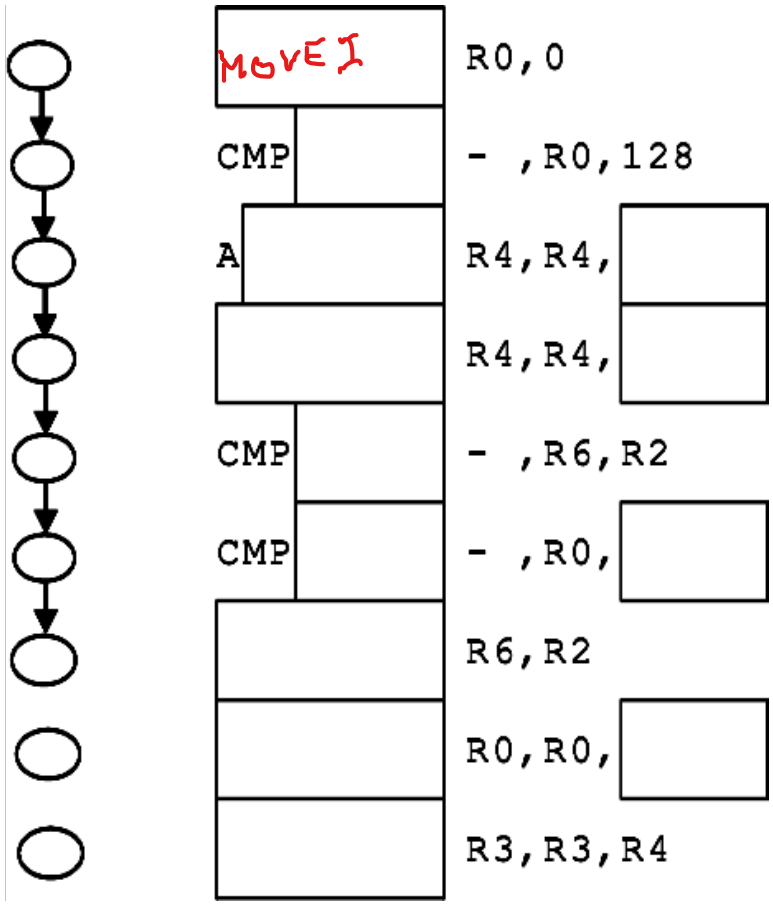
Exercicis de completar graf d'estats UC

- Cal completar:
  - Mnemotècnics
  - Transicions al graf d'estats
  - Etiquetes de les transicions
- Aspectes a considerar:
  - Descomposar codi C en accions que pugui realitzar la UPG en un cicle
  - A les comparacions,
    - Ens poden forçar qui ha de ser el primer o el segon operand
    - Hem de tenir en compte si les dades són naturals o enters
    - Cal determinar el valor del bit z correspon a TRUE/FALSE al codi font
  - A les accions, ens poden forçar algun caràcter del mnemotècnic
    - Per exemple, cal multiplicar per 2 amb mnemotècnic que comenci per `A`
    - `SHL R2, R2, +1` $\equiv$ `ADD R2, R2, R2`
    - Cal pensar en accions sinònimes

Parcial E3 Q1 16-17

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus natural.

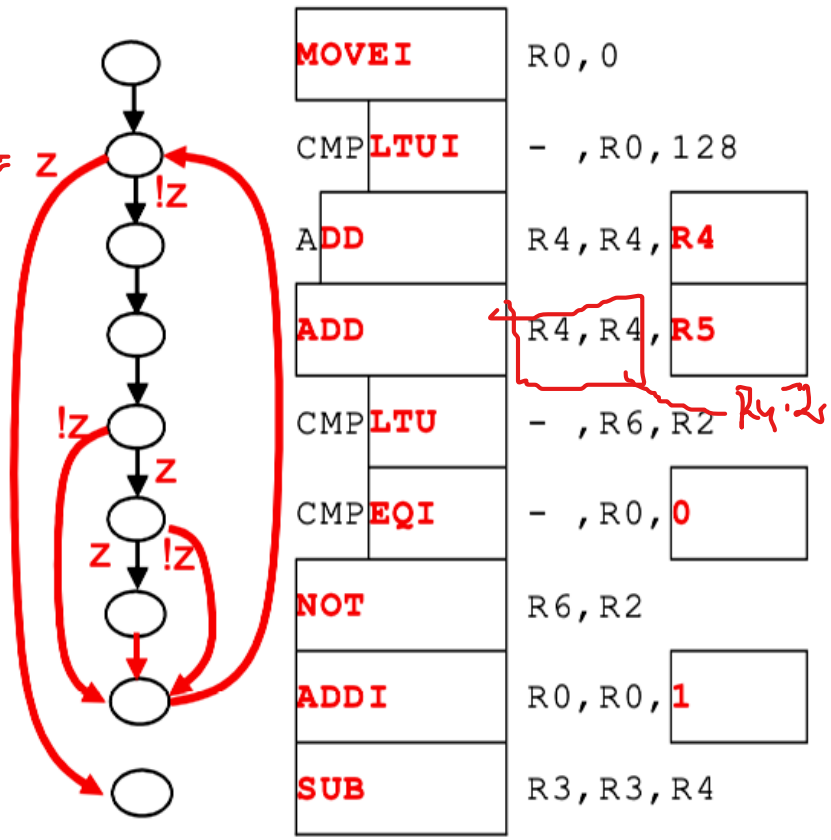
```
for (R0=0; R0<128; R0++) {
  R4=R4*2+R5;
  if ((R6>=R2) && (R0!=0))
    R6=not(R2)
}
R3=R3-R4;
```



Parcial E3 Q1 16-17

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus natural.

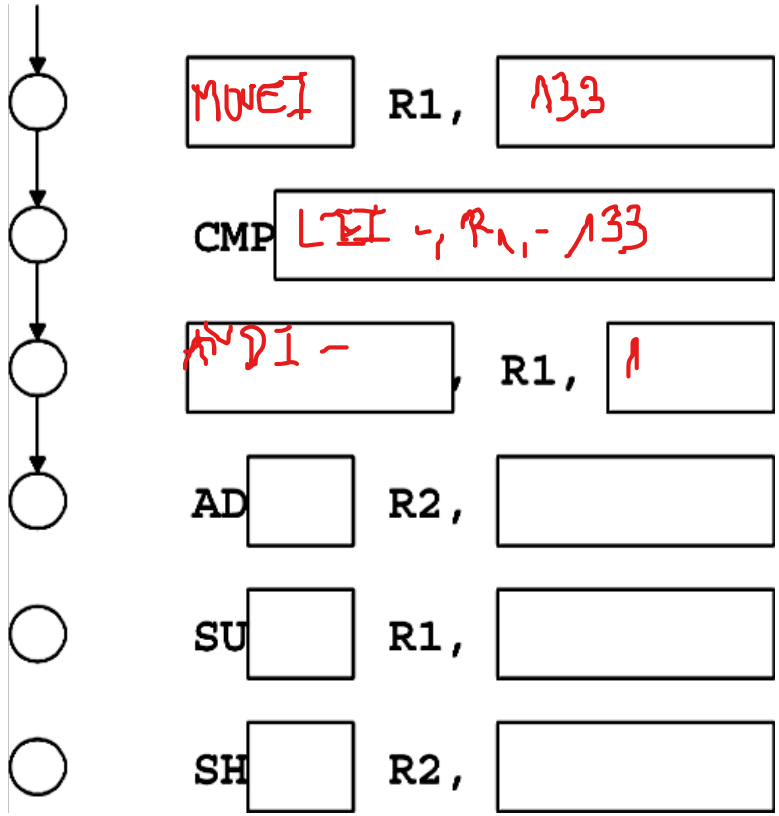
```
for (R0=0; R0<128; R0++) {
  R4=R4*2+R5;
  if ((R6>=R2) && (R0!=0))
    R6=not(R2)
}
R3=R3-R4;
```



Parcial E3 Q2 16-17

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus enter.

```
for(R1=133; R1>-133; R1--) {
  // R1<0> es el bit
  // de menys pes de R1
  if (R1<0>==1)
    R2=R2+R1
}
R2=R2/4;
```

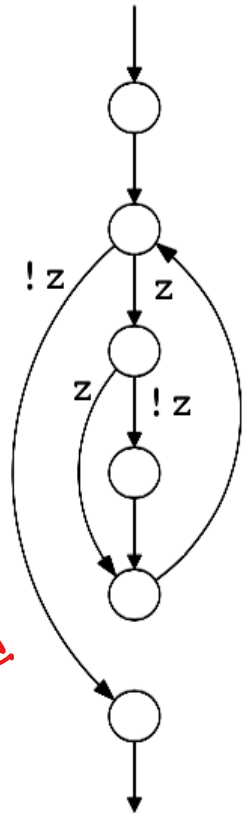


Parcial E3 Q2 16-17

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus enter.

```
for(R1=133; R1>-133; R1--) {
  // R1<0> es el bit
  // de menys pes de R1
  if (R1<0>==1)
    R2=R2+R1
}
R2=R2/4;
```

Sempre que pregunta per un bit fem un and



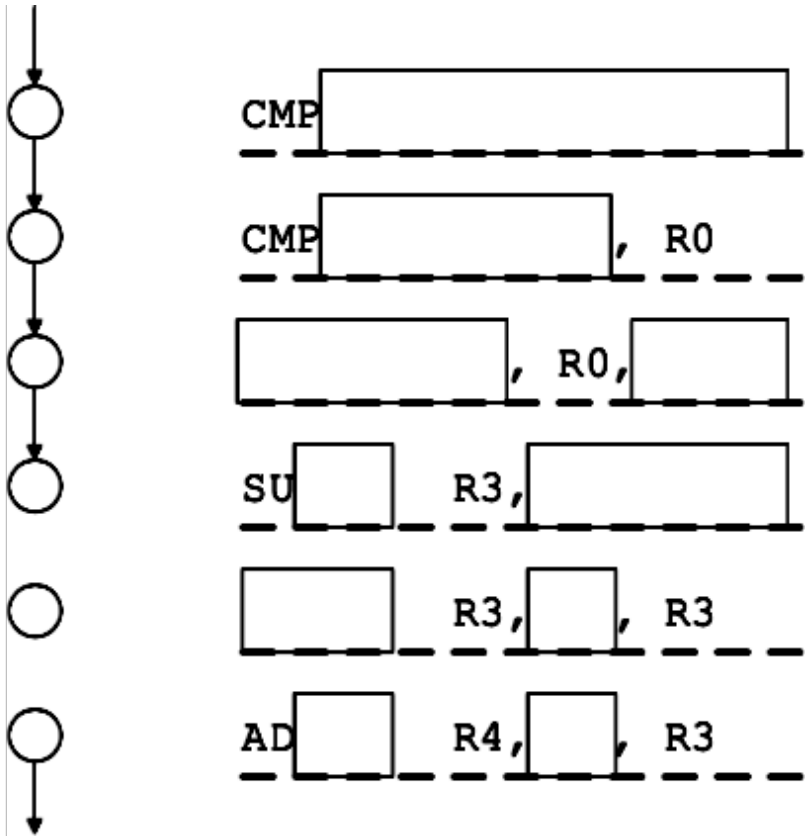

```
MOVEI  R1, 133
CMPLEI -, R1, -133
ANDI   -, R1, 1
ADD    R2, R2, R1
SUBI   R1, R1, 1
SHAI   R2, R2, -2
```

div per 4

Parcial E3 Q2 17-18

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus natural.

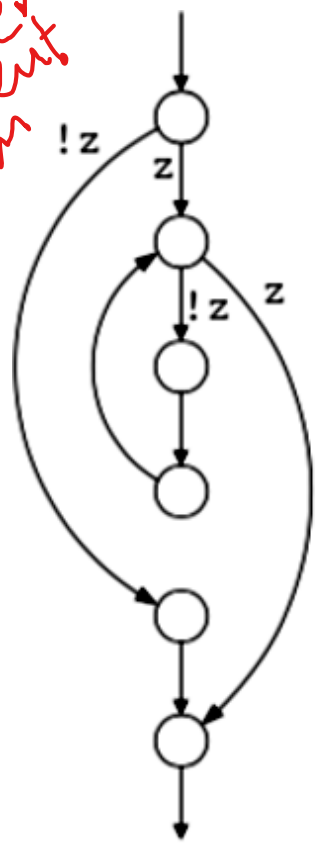
```
if (R3 > 50) {
  while (R0 >= R2) {
    R0 = R0 / 2;
    R3 = R3 - 1;
  }
}
else {
  // | es OR bit a bit
  R3 = R2 | R3;
}
R4 = R0 + R3;
```



Parcial E3 Q2 17-18

- Completeu (arcs, etiquetes i mnemotècnics) el fragment graf d'estats de la UC perquè, juntament amb la UPG, implementi el fragment de codi indicat.
- Les dades són de tipus natural.

```
if (R3 > 50) {
  while (R0 >= R2) {
    R0 = R0 / 2;
    R3 = R3 - 1;
  }
}
else {
  // | es OR bit a bit
  R3 = R2 | R3;
}
R4 = R0 + R3;
```




```
CMPLEUI  -, R3, 50
CMPLEU   -, R2, R0
SHLI     R0, R0, -1
SUBI     R3, R3, 1
OR       R3, R2, R3
ADD      R4, R0, R3
```

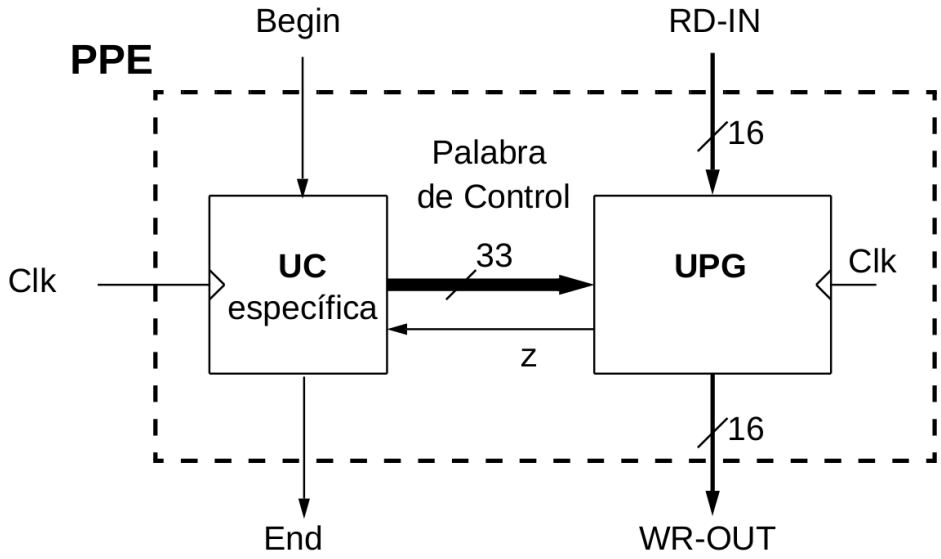
Exercicis a entregar a Atenea

- Enunciat disponible a Atenea
  - `https://atenea.upc.edu/pluginfile.php/3603442/mod_assign/introattachment/0/Tema%208%20-%20Exercicis%20en%20paper.pdf?forcedownload=1`
- Entrega a Atenea fins el dilluns 16/11
  - 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

- Esquema UC+UPG amb entrada/sortida



- Donat un problema, resoldre'l amb UPG+UC pot trigar més cicles i tenir un T_p major que si s'hagués resolt amb una UP específica
  - Però podrem reutilitzar la UPG per a d'altres problemes
  - I el cost de desenvolupament serà menor que utilitzat UP específica
- A alguns casos, la UPG rebrà dades de l'exterior (bus RD-IN) a un ritme d'una dada cada N cicles
  - Pot ser necessari utilitzar l'acció `NOP` per garantir que cada iteració del bucle triga exactament N cicles
- No oblideu realitzar el qüestionari ET8b i exercicis en paper (slide 39)

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.victorinox.com/global/en/Products/Swiss-Army-Knives/Medium-Pocket-Knives/Explorer/p/1.6703.

Introducció als Computadors

Tema 8: Unitat de Procés General (UPG)

http://personals.ac.upc.edu/enricm/Docencia/IC/IC8b.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.