

1. Els petits canvis són poderosos!

Construir una nova MS que serà molt igual a la MS1, tinguent en compte que disposem d'una memòria RAM de 128 caselles i que el tamany de la casella és de **1 byte**. Cal poguer executar totes les instruccions de la MS1, és a dir, ADD, CMP, MOV i BEQ. No modificar el format d'instrucció (ha de ser de 16 bits). No modificar la mida dels operants (han de ser de 16 bits).

Es demana:

- Dibuixa amb molt detall la nova UP (afegeix el hw que faci falta)
- Fer el diagrama d'estats simplificat indicant clarament què passa en cada estat i els senyals que provoquen les transicions d'un estat a un altre.
- Anomena tots els senyals de control que ha de suministrar la UC
- Escriu el següent programa en codi màquina de la nova MS a partir de la posició 0, a la memòria d'aquesta màquina:

```
inici: MOV A,B
      ADD B,B
      BEQ inici
```

tinguent en compte que les etiquetes tindran els següents valors: inici=0, a=100, b=102

- Compara la velocitat d'execució de les instruccions en la nova MS i en la MS1. Es pot executar un programa de 70 instruccions en la nova MS. Justifica la resposta.

3. Another time c'2 ?

Volem realitzar una modificació FIRMWARE a la MS1, per que pugui realitzar operacions aritmètiques amb signe. Hem decidit codificar els números en complement a 2 (C'2). Les noves instruccions de la MS seran:

SIGN F,D	$(D) \leftarrow C'2(F)$
ADD F,D	$(D) \leftarrow (F) + (D); \quad FZ \leftarrow (D) + (F) = 0$
CMP F,D	$FZ \leftarrow (D) = (F)$
BEQ D	Saltar a l'instrucció D \leftarrow FZ=1

Al registre A se li han acoblat 3 noves senyals de control:

AllSetA	Ompler tot el registre A de 1's
ResetA	Ompler tot el registre A de 0's
SetA	Posa el bit de menys pes de A a 1 i la resta a 0's.

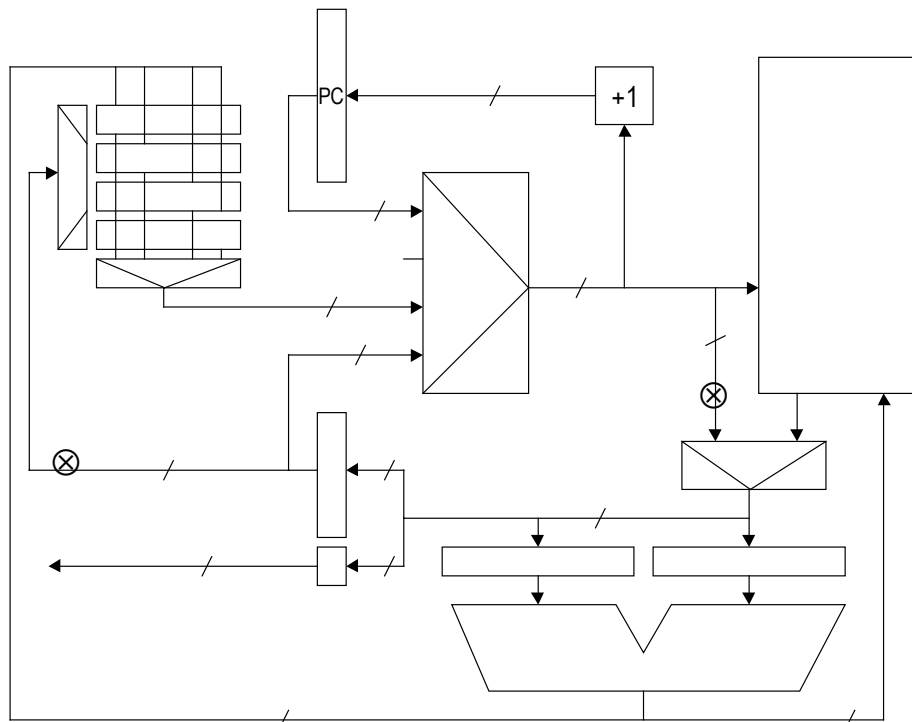
Es demana:

1. Dibuixa el graf d'estats simplificat indicant clarament les senyals de control que provoquen les transicions entre els diferents estats.
2. Especifica què passa en cada estat.

5. MAQUINA MAL ADREÇADA

Tornem a modificar la MS1 perquè funcioni amb instruccions d'un sol operand. Les instruccions són les de l'altra vegada: ADD, CMP, MOV, BEQ i LDB (LDB carrega el registre B de la ALU, mentre que ADD i CMP carreguen el registre A). La primera novetat és tenim *tres* modes d'adreçament: immediat, registre i registre indirecte (equivalents als del VAX). NO TENIM ADREÇAMENT SIMPLE. Per aquests dos últims es disposa d'un banc de quatre registres (R0, R1, R2 i R3).

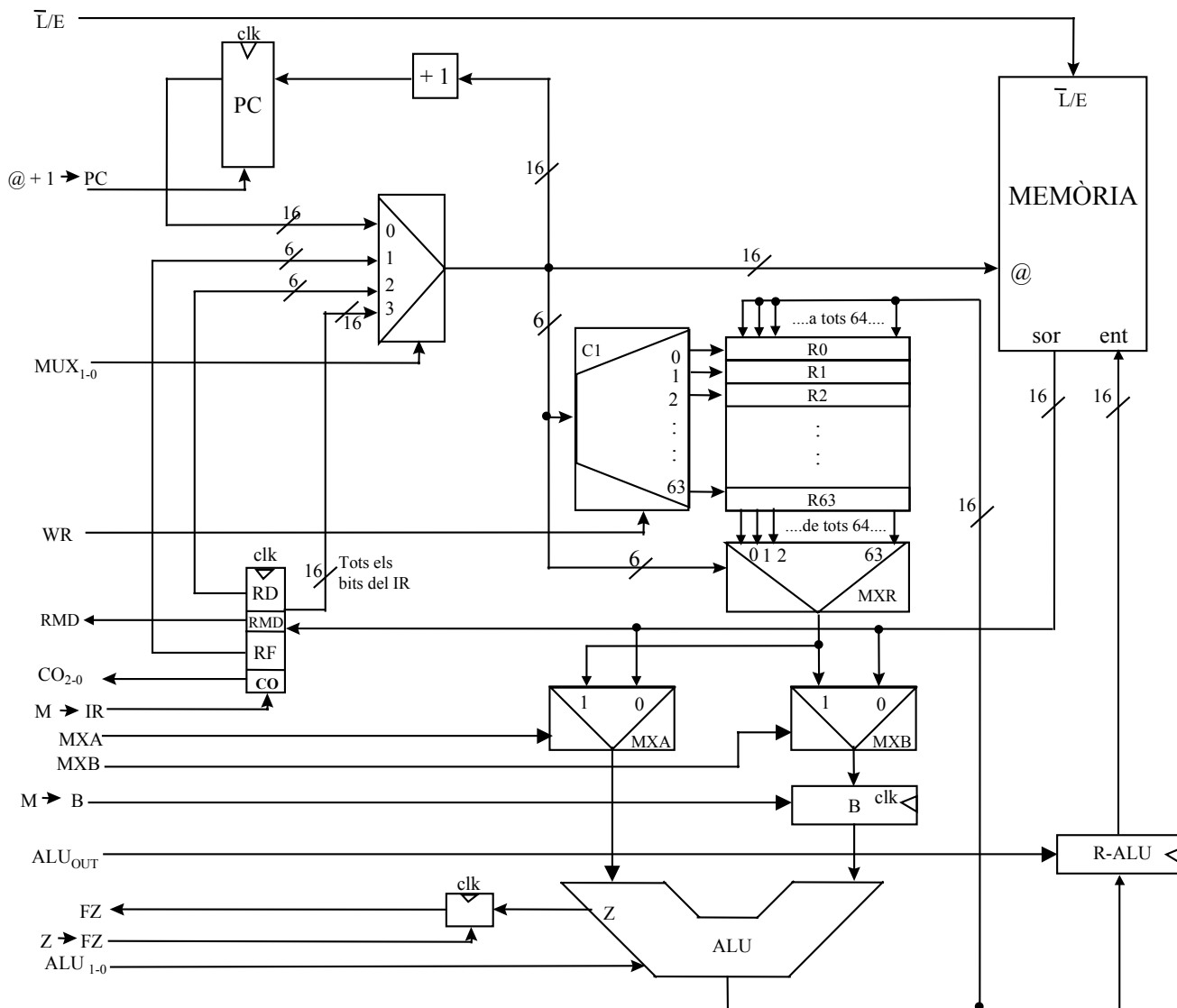
Per tal de facilitar l'adreçament per registre indirecte dividim l'IR en dues parts que es carregaran independentment: IRH, amb els bits del codi d'operació i el mode d'adreçament i IRL, amb l'operand de la instrucció.



- [1 punt] Especifica amb detall el format d'instrucció.
- [1 punt] Per a què serveixen les línies marcades amb una \otimes ?
- [1 punt] Determina i raona breument la amplada en bits dels camins i dels nous registres de la UP.
- [4 punts] Fes el graf d'estats *simplicat* indicant què passa a cada estat així com els senyals que determinen cada transició. NO FACIS LA TAULA DE SORTIDES.
- [3 punts] Fes un programa per a aquesta màquina que llegeixi la memòria desde l'adreça 1001 a la 2000 i posi l'adreça que contingui el número més gran a l'adreça 1000.

6. Registres per un tub!

Amb la finalitat de facilitar-vos les pràctiques s'està pensant en una màquina que disposi de 64 registres interns. Aquesta nova màquina executa les instruccions de la MS-1 (ADD, CMP, MOV i BEQ), a més a més d'una nova instrucció de 2 operands: XCHG. La màquina treballa amb dos modes d'adreçament diferents: registre directe i memòria directe, amb la particularitat que no totes les combinacions són possibles. Així doncs l'operand font sempre es troba en un registre. L'operand destí, en canvi, pot ser que es trobi en un registre o a la memòria. Fixeu-vos que les adreces de memòria són de 16 bits cosa que fa que hi hagi instruccions que seran de 16 bits i instruccions que seran de 32 bits segons si l'operand destí es troba en un registre a la memòria, respectivament. L'arquitectura proposada és la següent:



IMPORTANT:

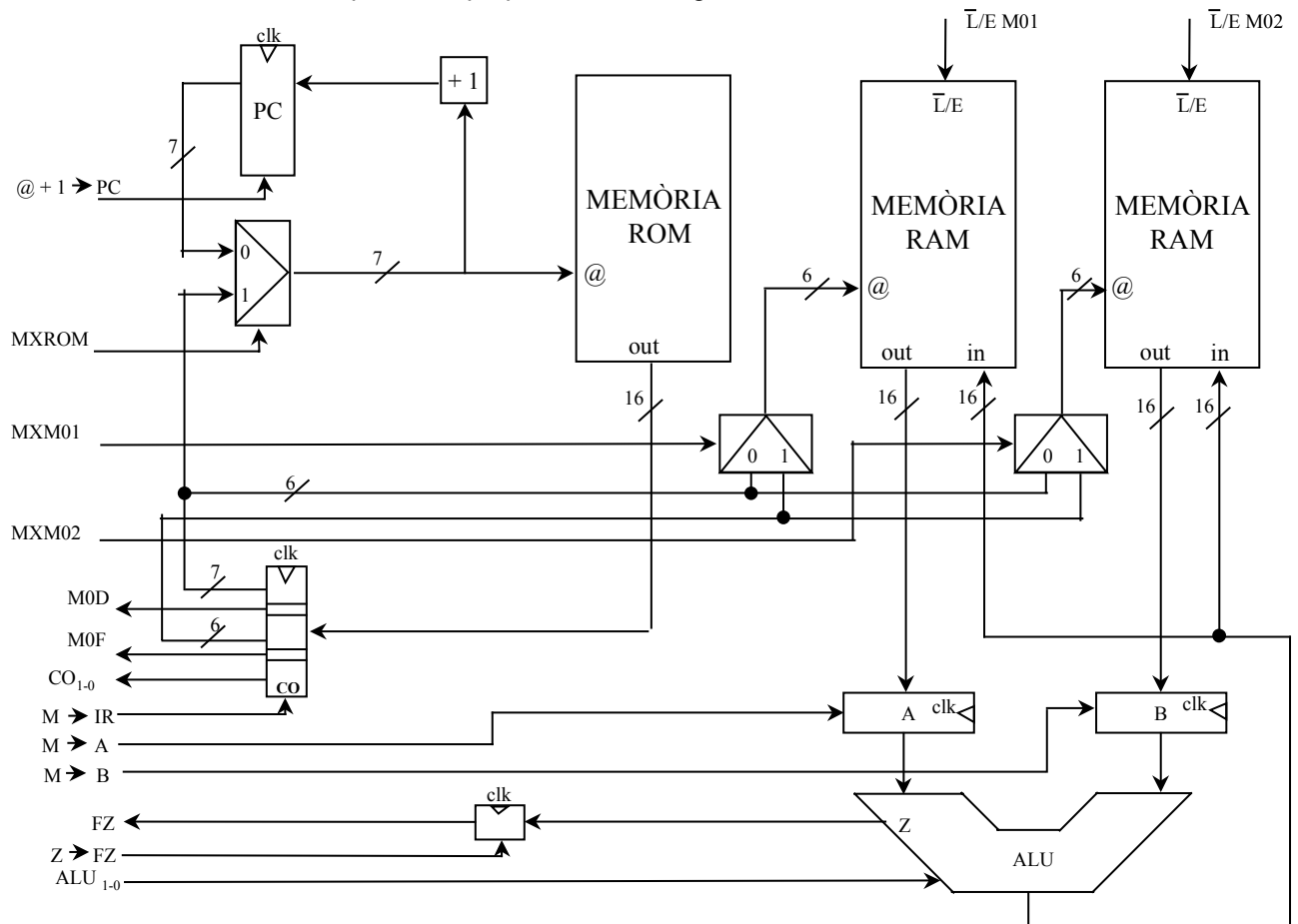
- Per les instruccions ADD, CMP i MOV, l'operand font sempre estarà a registre i el destí pot estar a registre (bit RMD=1) o a memòria (bit RMD=0).
- Per la instrucció BEQ l'operand destí sempre es trobarà a memòria.
- Per la instrucció XCHG tan l'operand font com el destí sempre es trobaran a registres i mai a la memòria.
- La instrucció XCHG el que ha de fer és intercanviar el contingut de dos registres.

Es demana:

- [3pts] Dissena el nou format d'instrucció de forma que contempli tots els casos.
- [5pts] Fer el diagrama d'estats simplificat indicant clarament què passa en cada estat i els senyals que provoquen les transicions d'un estat a un altre.
- [2pts] Proposa una modificació hardware de la màquina per tal de permetre que la instrucció XCHG també pugui tenir l'operand destí a la memòria.

7. La TRI-Memòria

Amb la finalitat d'aconseguir més rapidesa i més eficiència en l'execució d'instruccions (ADD, CMP, MOV i BEQ), hem organitzat l'emmagatzament de les dades i del programa en dos blocs. Un primer bloc de 128 caselles de memòria ROM, on només emmagatzemarem programa. Un segon bloc, format per dues memòries RAM de 64 caselles cadascuna, on es guarden les dades. Un dels objectius en el disseny passa per optimitzar al màxim els cicles per executar cada instrucció. L'arquitectura proposada és la següent:



IMPORTANT:

- Per les instruccions ADD, CMP i MOV, els operands poden estar situats a totes dues memòries RAM, és a dir, el primer operand pot estar situat a M01 o M02, al igual que el segon operand que també pot estar situat tan a M01 o M02.
 - Quan el bit MOF=0 llavors l'operand font es troba a la memòria M01
 - Quan el bit MOF=1 llavors l'operand font es troba a la memòria M02
 - Quan el bit MOD=0 llavors l'operand destí es troba a la memòria M01
 - Quan el bit MOD=1 llavors l'operand destí es troba a la memòria M02
- Quan els operands font i destí estiguin situats a una mateixa memòria, es pot utilitzar qualsevol posició de l'altre memòria per emmagatzemar temporalment la informació que calgui
- Considereu que la nova ALU fa les operacions aritmètico-lògiques que es necessitin (deixar passar transparentment el registre B, deixar passar transparentment el registre A)

Es demana:

- [2pts] Dissenya el nou format d'instrucció
- [5pts] Fer el diagrama d'estats simplificat indicant clarament què passa en cada estat i els senyals que provoquen les transicions d'un estat a un altre.
- [3pts] Proposa una modificació hardware de la màquina per tal d'arreglar el fet de que a vegades calgui modificar innecessàriament el contingut d'una posició de memòria per emmagatzemar informació temporal.

8. La Màquina Oberta

Volem fer una versió de la MS1 on totes les instruccions tinguin només un sol operand. Per aquelles que en necessiten dos afegim una cinquena instrucció, **LDB**, per carregar un valor al registre B de la ALU. Llavors per fer un *MOV F,D*, ara s'ha de fer un *LDB F* i, a continuació, un *MOV D*. Així doncs el conjunt d'instruccions d'aquesta màquina són:

- ADD D
- CMP D
- MOV D
- BEQ D
- LDB F

També es vol disposar de dos modes d'adreçament: el *simple* (el de sempre), quan a la instrucció conté l'adreça de l'operand; i l'*indirecte*, quan a la instrucció conté l'adreça on es troba l'adreça de l'operand. La instrucció BEQ només admetrà adreçament simple.

El format d'instrucció ha de ser de 16 bits, no deixant sense utilitzar cap bit.

- [2,5 pts] Dissenya i dibuixa la nova UP. Indicant la mida en bits de cada registre i camí de dades. Quin és el màxim de memòria (en bytes) que es pot adreçar?
- [1 pt] Descriu el nou format d'instrucció.
- [4 pts] Representa el graf d'estats simpificat d'aquesta màquina, indicant què passa en cada estat i en cada transició. NO FEU LA TAULA DE SORTIDES.
- [2,5 pts] Escriu un programa per a aquesta màquina que sumi els números de 1 al 1000 amb un bucle.

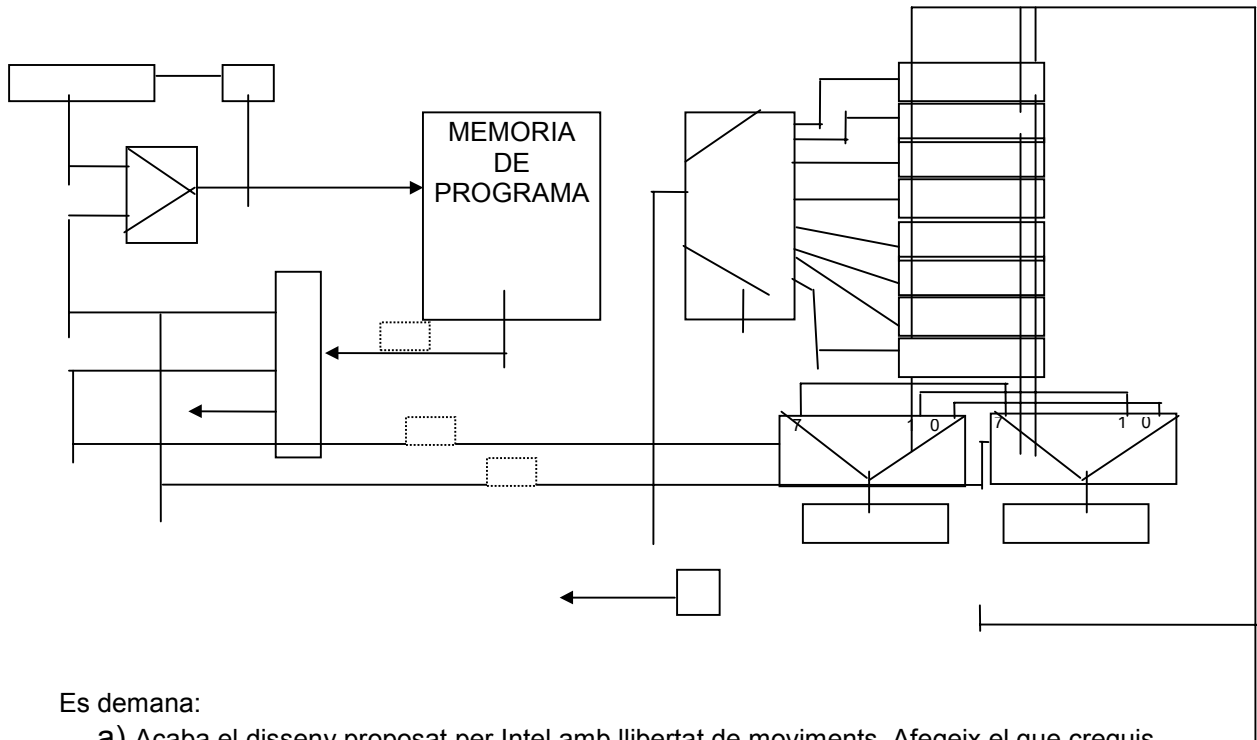
Sempre que sigui possible, justifica les teves respostes.

[4pts.]

10. Sense memòria

Amb la finalitat d'aconseguir més rapidesa i més eficiència en l'execució d'instruccions, hem organitzat l'emmagatzament de les dades i del programa en dos blocs. Un primer bloc de 128 caselles de memòria ROM, on només enmagatzemarem programa, i un segon de 8 caselles en un banc de registres, on guardarem les dades. Un dels objectius en el disseny passa per optimitzar al màxim els cicles per executar cada instrucció.

Un enginyer d'INTEL ha començat a dissenyar la nova arquitectura però ha quedat aturat a mig fer. De moment l'arquitectura proposada és la següent:



Es demana:

- Acaba el disseny proposat per Intel amb llibertat de moviments. Afegeix el que creguis que faci falta.
- Quin és el nou format d'instrucció de les instruccions ADD, CMP, MOV i BEQ per la nova Ms. Omple també les caselles puntajades de l'U.P. amb el valor corresponent.
- Dibuixa el graf d'estats simplificat, indica clarament els valors binàries que provoquen les transicions i les operacions que es realitzen en cada estat.
- Especifica els accessos a memòria que provoca el fet d'executar les instruccions ADD, CMP, MOV i BEQ.
- Prova de canviar el hw de la MS treguent els registres A i B. Dibuixa el graf d'estats simplificat de les instruccions ADD i MOV.

11. Maquina Senzi-QUÈ (2ona PART)

Construir una nova MS que sigui capaç d'executar el següent conjunt d'instruccions:

INSTRUCCIÓ	DESCRIPCIÓ
ADD Rf, #immediat, Rd	Rd <- Rf+immediat FZ <- (Rf+immediat=0)
ADD Rf1, Rf2, Rd	Rd <- Rf1+Rf2 FZ <- (Rf1+Rf2=0)
SUB Rf, #immediat, Rd	Rd <- Rf-immediat FZ <- (Rf-immediat=0)
SUB Rf1, Rf2, Rd	Rd <- Rf1-Rf2 FZ <- (Rf1-Rf2=0)
MOV Rf, Rd	Rd <- Rf FZ <- (Rf=0)
LOAD F, Rd	Rd <- (F)
STORE Rf, D	D <- Rf
BEQ D	si FZ=1 llavors IR <- (D) PC <- D+1

Les noves inst. utilitzen el m. registre, el m. immediat i el m. simple:

- En el mode registre l'operand es troba dins un registre del banc de registres. El banc de registres està format per 8 registres de 16 bits (R0..R7). Les etiquetes Rf,Rd,Rf1,Rf2 fan referencia a un dels 8 registres.
- En el m. immediat l'operand és un número de 5 bits codificat en binari natural que està contingut dins de la pròpia instrucció.
- En el m. simple l'operand es troba en una posició de memòria. F i D representen les posicions de memòria (són adreces) on es troben els operands.

IMPORTANT:

- La nova màquina ha de tenir una memòria RAM de 128 caselles i el tamany de la casella és de **2 bytes**.
- La memòria emmagatzema programes i dades.
- La nova màquina ha de tenir un banc de 8 registres de 16 bits per emmagatzemar dades.
- La mida dels operants ha de ser de 16 bits.
- Utilitzeu un registre IR de 16 bits.
- Considereu que teniu una ALU que fa les operacions aritmetico-lògiques que es necessitin.

Es demana:

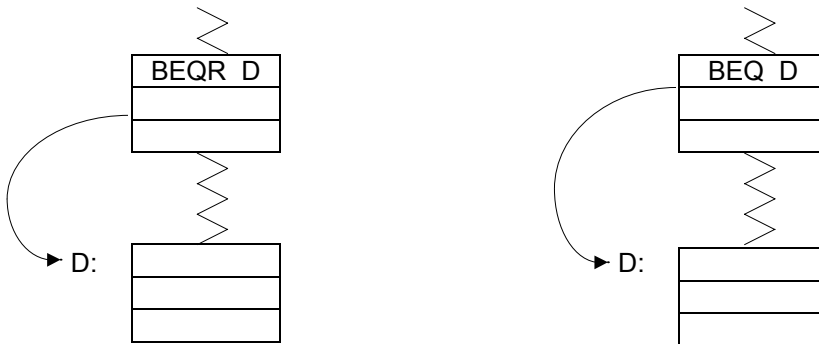
- a) [2pts.] Dissena el nou format d'instrucció
- b) [4pts.] Dibuixa amb molt detall la nova UP (afegeix el hw que faci falta)
- c) [4pts.] Fer el diagrama d'estats simplificat indicant clarament què passa en cada estat i els senyals que provoquen les transicions d'un estat a un altre.

[4pts.]

12.DELAY SLOTS

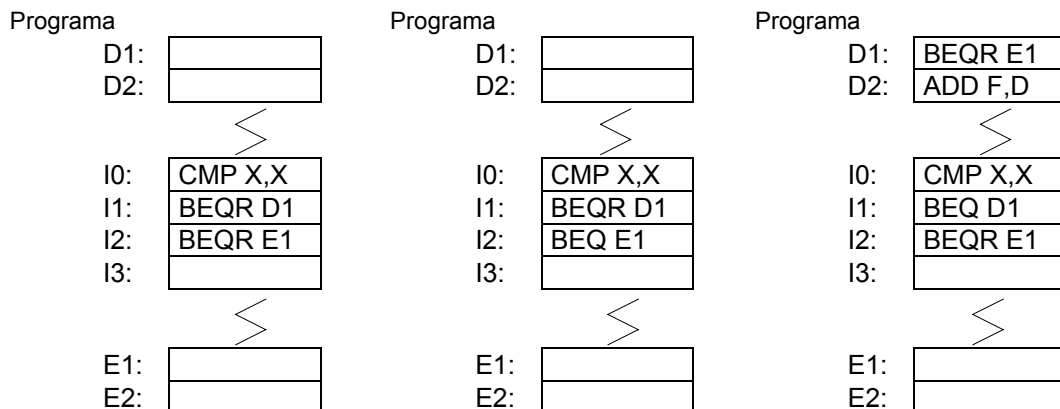
Sobre la Ms1 que hem vist a classe, volem substituir la instrucció MOV, per una nova instrucció de salt retardat que anomenarem BEQR. Aleshores la nova Ms disposarà de 4 instruccions que són ADD, CMP, BEQ i BEQR. Les 3 primeres instruccions es comporten com a la Ms1. La definició de l'instrucció BEQR és la següent:

BEQR D : Salta a l'instrucció apuntada per D, si FZ = 1 i després d'executar la següent instrucció a BEQR en seqüència. El flag es consulta en el moment d'executar el BEQR.



Es demana:

- (2 punts) Quina és la seqüència d'instruccions que s'executarien en els següents programes. Dóna la seqüència d'adreces, tinguent en compte que la primera instrucció que s'executa és la que està a l'adreça I0.



- (3 punts) Proposa l'arquitectura necessària per a permetre l'execució de les 4 instruccions esmentades. Fes el mínim de modificacions sobre l'arquitectura de la Ms1.
- (3 punts) Dibuixa el graf d'estats corresponent a les instruccions ADD, BEQ i BEQR, per l'arquitectura proposada en l'apartat anterior. Indica clarament els senyals que provoquen les transicions entre els diferents estats.
- (2 punts) Especifica què passa (vectors de sortida) en cada un dels estats dibuixats en l'apartat anterior.