Repàs: Examen final 2013-14 Q2

Joan Manuel Parcerisa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Pregunta 1. Rendiment i Consum

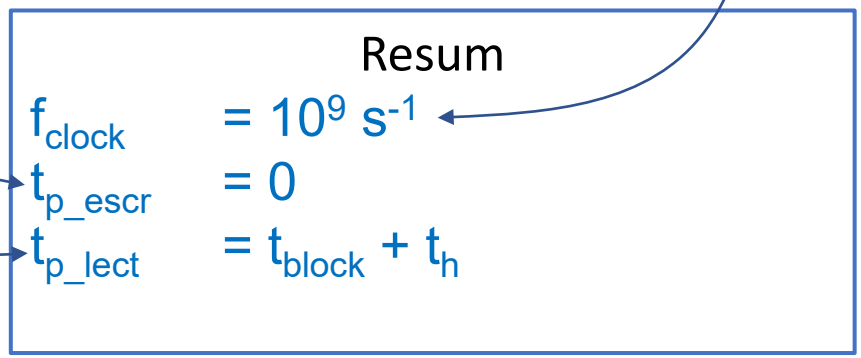
Un sistema computador treballa a una freqüència de rellotge de 1 GHz i disposa d'una memòria cache (MC) que utilitza una política d'escriptura immediata sense assignació. Els temps representatius del sistema de memòria són $t_h = 1$ cicle i $t_{block} = 149$ cicles.

En aquest computador s'executa un programa de 100 milions d'instruccions, les quals generen 150 milions de referències a memòria. S'ha observat que un 20% de les referències són d'escriptura i que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 2 s. i la potència dissipada és de 2 W.

Pregunta 1. Rendiment i Consum

Un sistema computador treballa a una freqüència de rellotge de 1 GHz i disposa d'una memòria cache (MC) que utilitza una política d'escriptura immediata sense assignació. Els temps representatius del sistema de memòria són t_h = 1 cicle i t_{block} = 149 cicles.

En aquest computador s'executa un programa de 100 milions d'instruccions, les quals generen 150 milions de referències a memòria. S'ha observat que un 20% de les referències són d'escriptura i que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 2 s. i la potència dissipada és de 2 W.



Pregunta 1. Rendiment i Consum

Un sistema computador treballa a una freqüència de rellotge de 1 GHz i disposa d'una memòria cache (MC) que utilitza una política d'escriptura immediata sense assignació. Els temps representatius del sistema de memòria són t_h = 1 cicle i t_{block} = 149 cicles.

En aquest computador s'executa un programa de 100 milions d'instruccions, les quals generen 150 milions de referències a memòria. S'ha observat que un 20% de les referències són d'escriptura i que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 2 s. i la potència dissipada és de 2 W.

Resum

$f_{clock} = 10^9\ s^{-1}$

$t_{p\_escr} = 0$

$t_{p\_lect} = t_{block} + t_h$

$t_h = 1$ cicle

$t_{block} = 149$ cicles

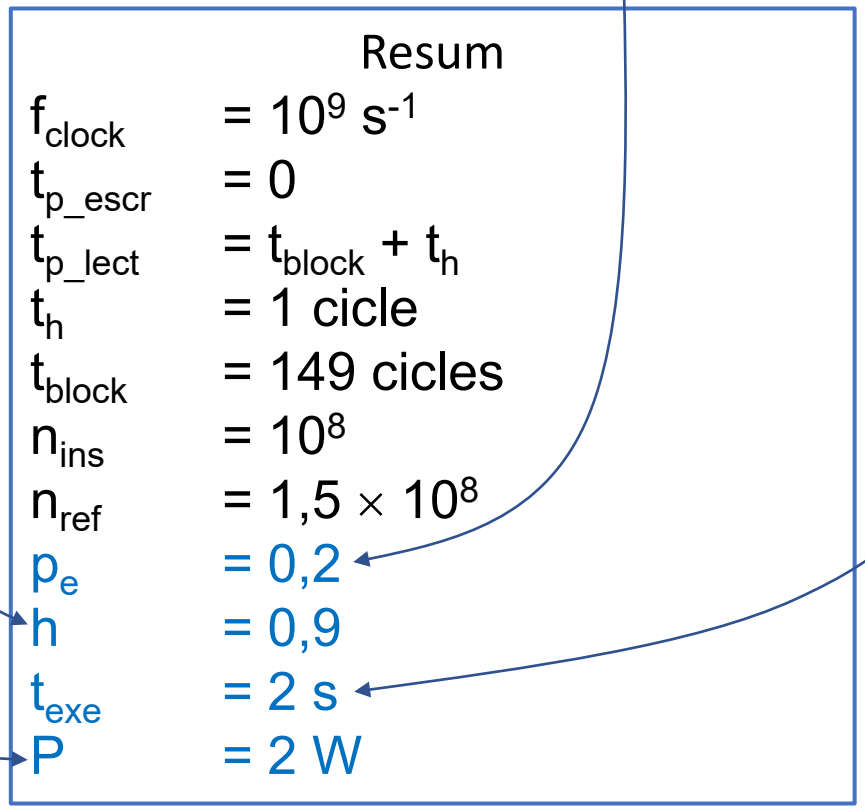
$n_{ins} = 10^8$

$n_{ref} = 1{,}5 \times 10^8$

Pregunta 1. Rendiment i Consum

Un sistema computador treballa a una freqüència de rellotge de 1 GHz i disposa d'una memòria cache (MC) que utilitza una política d'escriptura immediata sense assignació. Els temps representatius del sistema de memòria són $t_h = 1$ cicle i $t_{block} = 149$ cicles.

En aquest computador s'executa un programa de 100 milions d'instruccions, les quals generen 150 milions de referències a memòria. S'ha observat que un 20% de les referències són d'escriptura i que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 2 s. i la potència dissipada és de 2 W.



Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

- $CPI_{ideal} = CPI - CPI_{penal}$

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	= 1 cicle
t_{block}	= 149 cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	= 2 s
P	= 2 W

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$

$t_{exe} = CPI \cdot n_{ins} / f_{clock}$

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	$= 2$ s
P	$= 2$ W

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$
- $CPI_{penal} = m \cdot n_r \cdot t_p$

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	$= 2$ s
P	$= 2$ W

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$
- $CPI_{penal} = m \cdot n_r \cdot t_p$

$= (1-h) \cdot (n_{ref}/n_{ins}) \cdot [p_e \cdot t_{p\_escr} + (1-p_e) \cdot t_{p\_lect}]$

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	$= 2$ s
P	$= 2$ W

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$
- $CPI_{penal} = m \cdot n_r \cdot t_p$

$= (1-h) \cdot (n_{ref}/n_{ins}) \cdot [p_e \cdot t_{p\_escr} + (1-p_e) \cdot t_{p\_lect}]$

$= (1-0,9) \cdot (1,5 \cdot 10^8/10^8) \cdot [0 + (1-0,2) \cdot (149+1)]$

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1,5 \times 10^8$
p_e	$= 0,2$
h	$= 0,9$
t_{exe}	$= 2$ s
P	$= 2$ W

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	$= 2$ s
P	$= 2$ W

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$
- $CPI_{penal} = m \cdot n_r \cdot t_p$

$= (1-h) \cdot (n_{ref}/n_{ins}) \cdot [p_e \cdot t_{p\_escr} + (1-p_e) \cdot t_{p\_lect}]$

$= (1-0{,}9) \cdot (1{,}5 \cdot 10^8/10^8) \cdot [0 + (1-0{,}2) \cdot (149+1)]$

$= 0{,}1 \cdot 1{,}5 \cdot 0{,}8 \cdot 150 = 18$

Pregunta 1. Rendiment i Consum

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

f_{clock}	$= 10^9\ s^{-1}$
t_{p_escr}	$= 0$
t_{p_lect}	$= t_{block} + t_h$
t_h	$= 1$ cicle
t_{block}	$= 149$ cicles
n_{ins}	$= 10^8$
n_{ref}	$= 1{,}5 \times 10^8$
p_e	$= 0{,}2$
h	$= 0{,}9$
t_{exe}	$= 2$ s
P	$= 2$ W

- $CPI_{ideal} = CPI - CPI_{penal}$
- $CPI = t_{exe} \cdot f_{clock}/n_{ins} = 2 \cdot 10^9/10^8 = 20$
- $CPI_{penal} = m \cdot n_r \cdot t_p$

  $= (1-h) \cdot (n_{ref}/n_{ins}) \cdot [p_e \cdot t_{p\_escr} + (1-p_e) \cdot t_{p\_lect}]$

  $= (1-0{,}9) \cdot (1{,}5 \cdot 10^8/10^8) \cdot [0 + (1-0{,}2) \cdot (149+1)]$

  $= 0{,}1 \cdot 1{,}5 \cdot 0{,}8 \cdot 150 = 18$

- $\mathbf{CPI_{ideal}} = 20 - 18 = \mathbf{2}$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$t_0 = CPI \cdot n_{ins} / f_0$

f_0	$= 1 \cdot 10^9\ s^{-1}$
f_1	$= 2 \cdot 10^9\ s^{-1}$
t_0	$= 2\ s$
P_0	$= 2\ W$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$

f_0	$= 1 \cdot 10^9\ s^{-1}$
f_1	$= 2 \cdot 10^9\ s^{-1}$
t_0	$= 2\ s$
P_0	$= 2\ W$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$t_0 = \mathbf{CPI} \cdot n_{ins} / f_0$

$\mathbf{t_1} = \mathbf{CPI} \cdot n_{ins} / f_1$

$= (t_0 \cdot f_0) / f_1$

$f_0 = 1 \cdot 10^9 \ s^{-1}$

$f_1 = 2 \cdot 10^9 \ s^{-1}$

$t_0 = 2 \ s$

$P_0 = 2 \ W$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

f_0	$= 1 \cdot 10^9\ s^{-1}$
f_1	$= 2 \cdot 10^9\ s^{-1}$
t_0	$= 2\ s$
P_0	$= 2\ W$

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$
$= (t_0 \cdot f_0) / f_1$
$= (2 \cdot 10^9) / 2 \cdot 10^9$
$= \mathbf{1\ s}$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$f_0 = 1 \cdot 10^9 \ s^{-1}$
$f_1 = 2 \cdot 10^9 \ s^{-1}$
$t_0 = 2 \ s$
$P_0 = 2 \ W$

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$
$= (t_0 \cdot f_0) / f_1$
$= (2 \cdot 10^9) / 2 \cdot 10^9$
$= \mathbf{1\ s}$

$P_0 = \alpha \cdot C \cdot V^2 \cdot f_0$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$f_0 = 1 \cdot 10^9\ s^{-1}$
$f_1 = 2 \cdot 10^9\ s^{-1}$
$t_0 = 2\ s$
$P_0 = 2\ W$

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$
$= (t_0 \cdot f_0) / f_1$
$= (2 \cdot 10^9) / 2 \cdot 10^9$
$= \mathbf{1\ s}$

$P_0 = \alpha \cdot C \cdot V^2 \cdot f_0$

$\mathbf{P_1} = \alpha \cdot C \cdot V^2 \cdot f_1$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

$f_0 = 1 \cdot 10^9\ s^{-1}$
$f_1 = 2 \cdot 10^9\ s^{-1}$
$t_0 = 2\ s$
$P_0 = 2\ W$

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$
$= (t_0 \cdot f_0) / f_1$
$= (2 \cdot 10^9) / 2 \cdot 10^9$
$= \mathbf{1\ s}$

$P_0 = \alpha \cdot C \cdot V^2 \cdot f_0$

$\mathbf{P_1} = \alpha \cdot C \cdot V^2 \cdot f_1$
$= (P_0 / f_0) \cdot f_1$

Pregunta 1. Rendiment i Consum

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

f_0	$= 1 \cdot 10^9\ s^{-1}$
f_1	$= 2 \cdot 10^9\ s^{-1}$
t_0	$= 2\ s$
P_0	$= 2\ W$

$t_0 = CPI \cdot n_{ins} / f_0$

$\mathbf{t_1} = CPI \cdot n_{ins} / f_1$
$= (t_0 \cdot f_0) / f_1$
$= (2 \cdot 10^9) / 2 \cdot 10^9$
$= \mathbf{1\ s}$

$P_0 = \alpha \cdot \mathbf{C} \cdot \mathbf{V^2} \cdot f_0$

$\mathbf{P_1} = \alpha \cdot \mathbf{C} \cdot \mathbf{V^2} \cdot f_1$
$= (P_0 / f_0) \cdot f_1$
$= (2 / 10^9) \cdot 2 \cdot 10^9$
$= \mathbf{4\ w}$

Pregunta 2. Memòria Cache

Considera un sistema format per un processador amb adreces de 16 bits i una memòria cache (MC) amb la següent configuració:

- capacitat total: 4 blocs
- mida del bloc: 16 bytes
- correspondència directa
- escriptura retardada amb assignació

En un moment donat l'estat de la part de control de la MC és:

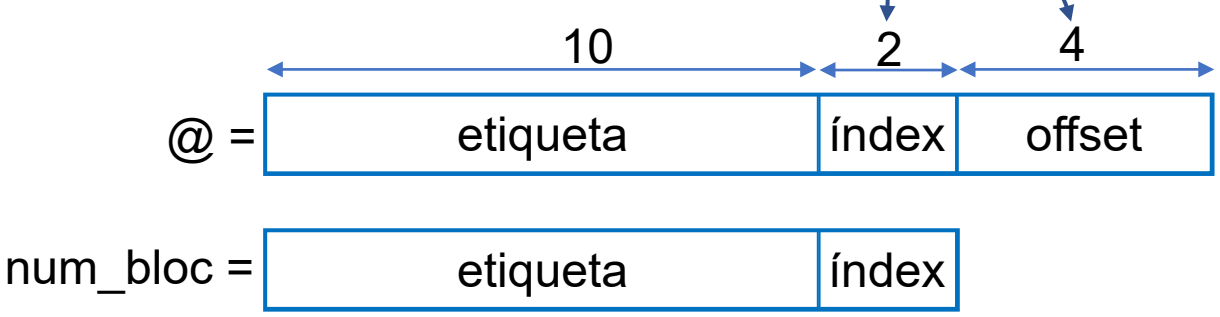
índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

Pregunta 2. Memòria Cache

a) Indica (en hexadecimal) els números de bloc de memòria principal (MP) que hi ha guardats a la MC.

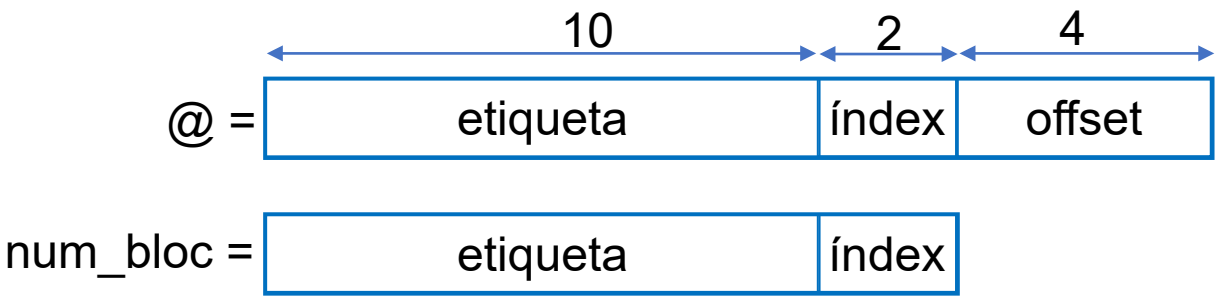
Considera un sistema format per un processador amb adreces de 16 bits i una memòria cache (MC) amb la següent configuració:

- capacitat total: 4 blocs
- mida del bloc: 16 bytes
- correspondència directa



Pregunta 2. Memòria Cache

a) Indica (en hexadecimal) els números de bloc de memòria principal (MP) que hi ha guardats a la MC.



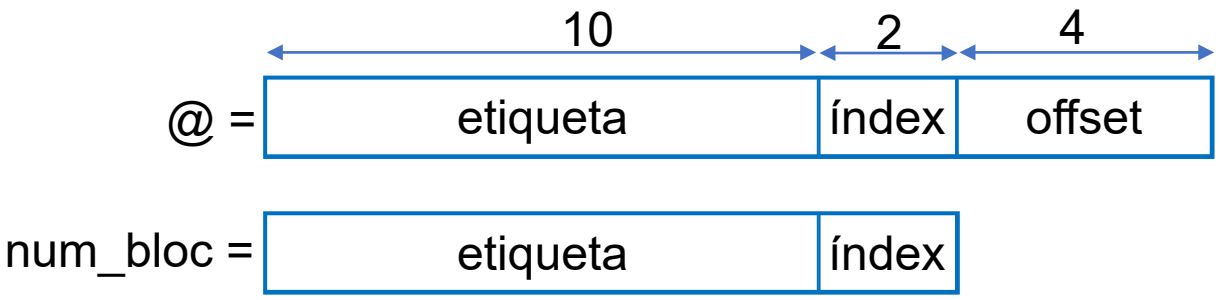
índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

⇨

etiqueta	índex	num_bloc (hex)

Pregunta 2. Memòria Cache

a) Indica (en hexadecimal) els números de bloc de memòria principal (MP) que hi ha guardats a la MC.



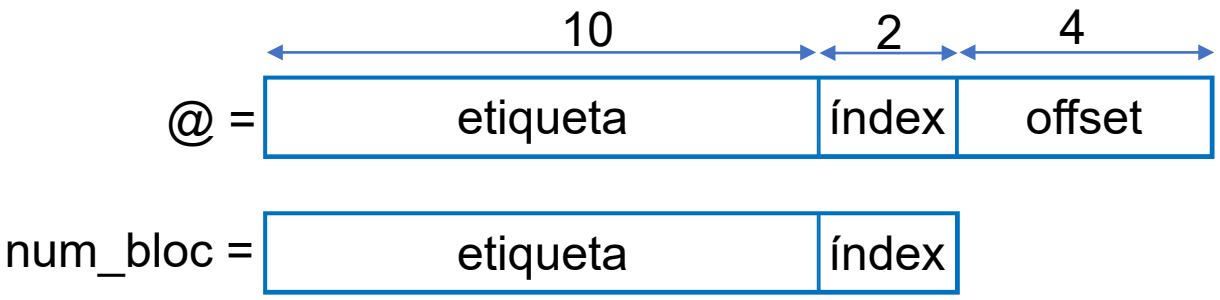
índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

⇨

etiqueta	índex	num_bloc (hex)
0000 0000 01	00	
0000 0000 00	01	
0000 0000 10	10	
0000 0000 11	11	

Pregunta 2. Memòria Cache

a) Indica (en hexadecimal) els números de bloc de memòria principal (MP) que hi ha guardats a la MC.



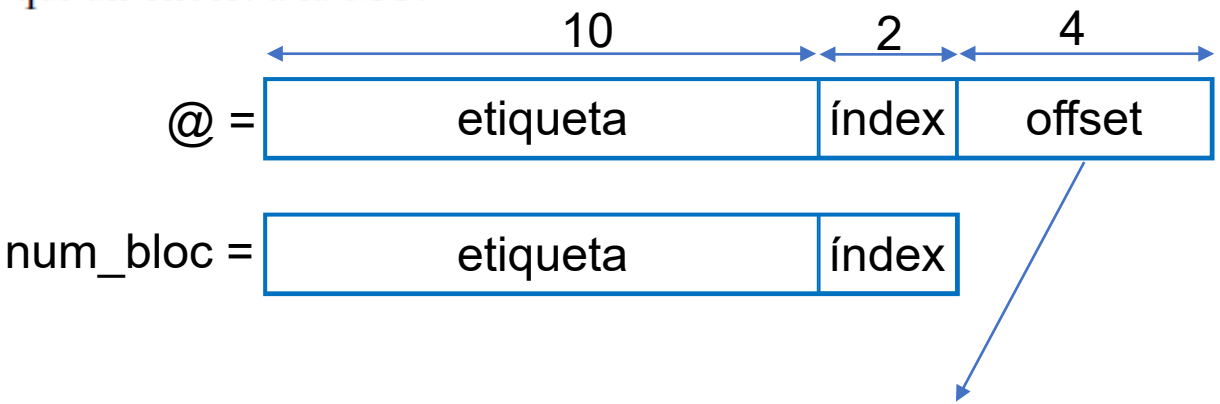
índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

⇨

etiqueta	índex	num_bloc (hex)
0000 0000 01	00	**0x004**
0000 0000 00	01	**0x001**
0000 0000 10	10	**0x00A**
0000 0000 11	11	**0x00F**

Pregunta 2. Memòria Cache

b) Indica una adreça de memòria (en hexadecimal) de la següent referència a tractar que provoqui un encert a la MC.

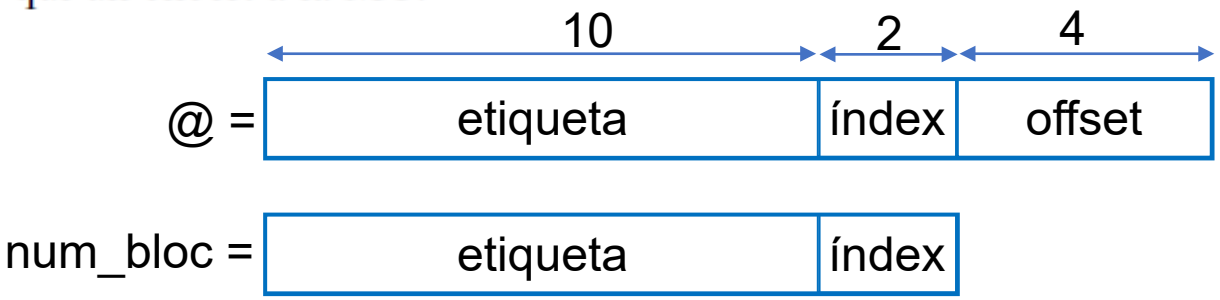


Sols cal afegir un offset qualsevol de 4 bits (dígit X = 0..F) a qualsevol dels números de bloc que tenim guardats:

num_bloc (hex)	adreces "hit" (hex)
0x004	
0x001	
0x00A	
0x00F	

Pregunta 2. Memòria Cache

b) Indica una adreça de memòria (en hexadecimal) de la següent referència a tractar que provoqui un encert a la MC.



Sols cal afegir un offset qualsevol de 4 bits (dígit X = 0..F) a qualsevol dels números de bloc que tenim guardats:

num_bloc (hex)	adreces "hit" (hex)
0x004	**0x004X**
0x001	**0x001X**
0x00A	**0x00AX**
0x00F	**0x00FX**

Per exemple: **0x0040**

Pregunta 2. Memòria Cache

c) Considera que la propera referència a tractar és una escriptura a l'adreça 0x0190. Indica quin serà l'estat final de la part de control de la MC després de tractar-la.

@ = 0x0190 = 0000 0001 1001 0000

etiqueta = 6

índex = 1

Pregunta 2. Memòria Cache

c) Considera que la propera referència a tractar és una escriptura a l'adreça 0x0190. Indica quin serà l'estat final de la part de control de la MC després de tractar-la.

@ = 0x0190 = 0000 0001 1001 0000

etiqueta = 6

índex = 1

índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

- Fallada d'escriptura (retardada amb assignació)

⇒ Llegim un bloc de MP i reemplacem el bloc de l'entrada 1 (que està Dirty i s'ha d'escriure a MP)

Pregunta 2. Memòria Cache

c) Considera que la propera referència a tractar és una escriptura a l'adreça 0x0190. Indica quin serà l'estat final de la part de control de la MC després de tractar-la.

@ = 0x0190 = 0000 0001 1001 0000

etiqueta = 6

índex = 1

índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

- Fallada d'escriptura (retardada amb assignació)

⇒ Llegim un bloc de MP i reemplacem el bloc de l'entrada 1 (que està Dirty i s'ha d'escriure a MP)

índex MC	V	D	etiqueta
0	1	1	1
1	**1**	**1**	**6**
2	1	0	2
3	1	0	3

⇒ Anotem: V=1, D=1, etiqueta = 6

Pregunta 2. Memòria Cache

Quants bytes es transfereixen de MP cap a MC durant la gestió d'aquesta referència?

Quants bytes es transfereixen de MC cap a MP durant la gestió d'aquesta referència?

Pregunta 2. Memòria Cache

Quants bytes es transfereixen de MP cap a MC durant la gestió d'aquesta referència?

1 bloc = 16 bytes

Quants bytes es transfereixen de MC cap a MP durant la gestió d'aquesta referència?

Pregunta 2. Memòria Cache

Quants bytes es transfereixen de MP cap a MC durant la gestió d'aquesta referència?

1 bloc = 16 bytes

Quants bytes es transfereixen de MC cap a MP durant la gestió d'aquesta referència?

1 bloc = 16 bytes (és el bloc reemplaçat, que estava "Dirty")

Pregunta 2. Memòria Cache

d) Considera ara que la MC és totalment associativa, amb reemplaçament LRU, i amb la mateixa capacitat total i mida de bloc. Sabent que les darreres referències que s'han tractat són: 0x000E, 0x00FE, 0x00AC, 0x000A, 0x0018 i 0x00CE, indica quines són les etiquetes (en hexadecimal) que hi ha guardades a la part de control de la MC.

Adreces: 0x000E, 0x00FE, 0x00AC, 0x000A, 0x0018, 0x00CE

Num_bloc: 0x000, 0x00F, 0x00A, 0x000, 0x001, 0x00C

Pregunta 2. Memòria Cache

d) Considera ara que la MC és totalment associativa, amb reemplaçament LRU, i amb la mateixa capacitat total i mida de bloc. Sabent que les darreres referències que s'han tractat són: 0x000E, 0x00FE, 0x00AC, 0x000A, 0x0018 i 0x00CE, indica quines són les etiquetes (en hexadecimal) que hi ha guardades a la part de control de la MC.

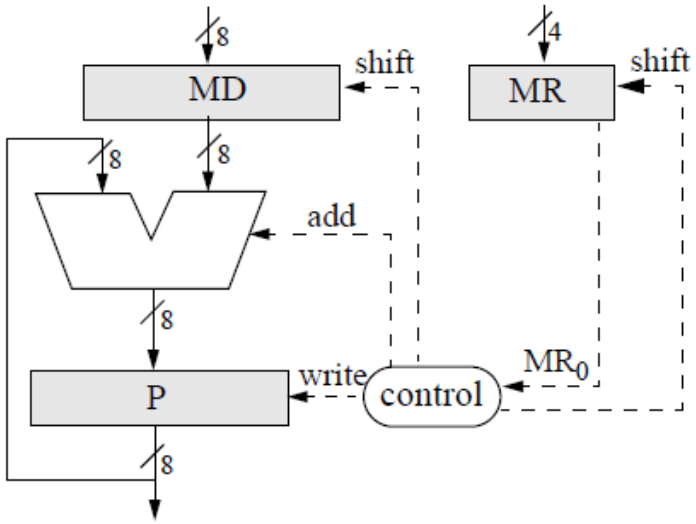
Adreces: 0x000E, 0x00FE, 0x00AC, 0x000A, 0x0018, 0x00CE

Num_bloc: 0x000, 0x00F, 0x00A, 0x000, 0x001, 0x00C

LRU: Quedaran en cache els 4 blocs més recents:

Pregunta 3. Multiplicació d'enters

Sigui el següent diagrama del multiplicador seqüencial de nombres naturals de 4 bits anàleg al que s'ha estudiat durant el curs, el qual calcula el producte amb 8 bits:



Suposem que amb aquest circuit multipliquem els números binaris de 4 bits 1001 (multiplicand) i 1011 (multiplicador). Completa la següent taula, que mostra els valors en binari dels registres P, MD, i MR després de la inicialització i després de cada iteració, afegint tantes iteracions com facin falta:

iter.	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
ini	0	0	0	0	0	0	0	0									1	0	1	1
1																				
2																				

Pregunta 3. Multiplicació d'enters

Inicialització

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1																				
2																				
3																				
4																				

Pregunta 3. Multiplicació d'enters

Iteració 1: $MR_0 = 1$

P = P + MD

P	0	0	0	0	0	0	0	0
+ MD	0	0	0	0	**1**	**0**	**0**	**1**
=	0	0	0	0	**1**	**0**	**0**	**1**

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**												
2																				
3																				
4																				

Pregunta 3. Multiplicació d'enters

Iteració 1: $MR_0 = 1$

P = P + MD

MD = MD << 1

MR = MR >> 1

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	0	**1**	**0**	**0**	**1**	0	0	**1**	**0**	**1**
2																				
3																				
4																				

Pregunta 3. Multiplicació d'enters

Iteració 2: $MR_0 = 1$

P = P + MD

P	0	0	0	0	**1**	**0**	**0**	**1**
+ MD	0	0	0	**1**	**0**	**0**	**1**	0
=	0	0	0	**1**	**1**	**0**	**1**	**1**

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	0	**1**	**0**	**0**	**1**	0	0	**1**	**0**	**1**
2	0	0	0	**1**	**1**	**0**	**1**	**1**												
3																				
4																				

Pregunta 3. Multiplicació d'enters

Iteració 2: $MR_0 = 1$

P = P + MD

MD = MD << 1

MR = MR >> 1

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	0	**1**	**0**	**0**	**1**	0	0	**1**	**0**	**1**
2	0	0	0	**1**	**1**	**0**	**1**	**1**	0	0	**1**	**0**	**0**	**1**	0	0	0	0	**1**	**0**
3																				
4																				

Pregunta 3. Multiplicació d'enters

Iteració 3: $MR_0 = 0$

MD = MD << 1
MR = MR >> 1

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	0	**1**	**0**	**0**	**1**	0	0	**1**	**0**	**1**
2	0	0	0	**1**	**1**	**0**	**1**	**1**	0	0	**1**	**0**	**0**	**1**	0	0	0	0	**1**	**0**
3	0	0	0	**1**	**1**	**0**	**1**	**1**	0	**1**	**0**	**0**	**1**	0	0	0	0	0	0	**1**
4																				

Pregunta 3. Multiplicació d'enters

Iteració 4: $MR_0 = 1$

$P = P + MD$ →

P	0	0	0	1	1	0	1	1
+ MD	0	1	0	0	1	0	0	0
=	0	1	1	0	0	0	1	1

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1
1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1
2	0	0	0	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1
4	0	1	1	0	0	0	1	1												

Pregunta 3. Multiplicació d'enters

Iteració 4: $MR_0 = 1$

P = P + MD

MD = MD << 1

MR = MR >> 1

P	0	0	0	**1**	**1**	**0**	**1**	**1**
MD	0	**1**	**0**	**0**	**1**	0	0	0
P'	0	**1**	**1**	**0**	**0**	**0**	**1**	**1**

iter	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
init	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	0	**1**	**0**	**0**	**1**	0	0	**1**	**0**	**1**
2	0	0	0	**1**	**1**	**0**	**1**	**1**	0	0	**1**	**0**	**0**	**1**	0	0	0	0	**1**	**0**
3	0	0	0	**1**	**1**	**0**	**1**	**1**	0	**1**	**0**	**0**	**1**	0	0	0	0	0	0	**1**
4	0	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	0	0	0		0	0	0	0

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

- Dades escalars en registres
  - `a`,`b`,`i` (però no `k`, que anirà a la pila!!)

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

- Dades escalars en registres
  - `a,b,i` (però no `k`, que anirà a la pila!!)
- Registres generats ABANS de la crida i usats DESPRÉS
  - `a ?`
  - `b ?`
  - `i ?`

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

a

- Dades escalars en registres
  - `a,b,i` (però no `k`, que anirà a la pila!!)
- Registres generats ABANS de la crida i usats DESPRÉS
  - `a` (es genera al principi i s'usa en cada iteració)
  - `b` ?
  - `i` ?

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

a, b

- Dades escalars en registres
  - `a,b,i` (però no `k`, que anirà a la pila!!)
- Registres generats ABANS de la crida i usats DESPRÉS
  - `a` (es genera al principi i s'usa en cada iteració)
  - `b` (es genera al principi i s'usa en cada iteració)
  - `i` ?

Pregunta 4. Subrutines

Donada la següent declaració de funcions en C:

```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

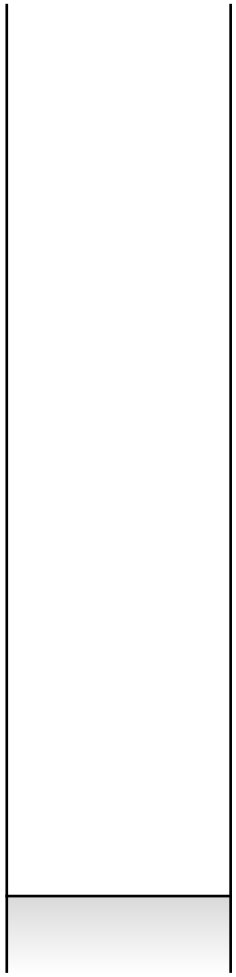
a, b, i

- Dades escalars en registres
  - `a,b,i` (però no `k`, que anirà a la pila!!)
- Registres generats ABANS de la crida i usats DESPRÉS
  - `a` (es genera al principi i s'usa en cada iteració)
  - `b` (es genera al principi i s'usa en cada iteració)
  - `i` (la sentència `i=i+1` després de cada crida usa el valor generat en l'anterior iteració)

Pregunta 4. Subrutines

Dibuixa el bloc d'activació de `subr1` (sobre el diagrama que tens a la dreta), especificant-hi la posició on apunta el registre `$sp` un cop reservat l'espai corresponent a la pila, així com el nom de cada registre i/o variable, i la seva posició (desplaçament relatiu al `$sp`).

```
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
              subr2(v, &a[3], &k);
}
```

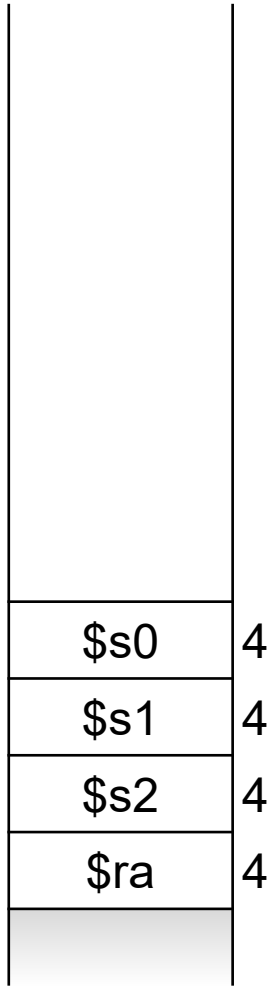


Pregunta 4. Subrutines

Dibuixa el bloc d'activació de `subr1` (sobre el diagrama que tens a la dreta), especificant-hi la posició on apunta el registre `$sp` un cop reservat l'espai corresponent a la pila, així com el nom de cada registre i/o variable, i la seva posició (desplaçament relatiu al `$sp`).

```
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
              subr2(v, &a[3], &k);
}
```

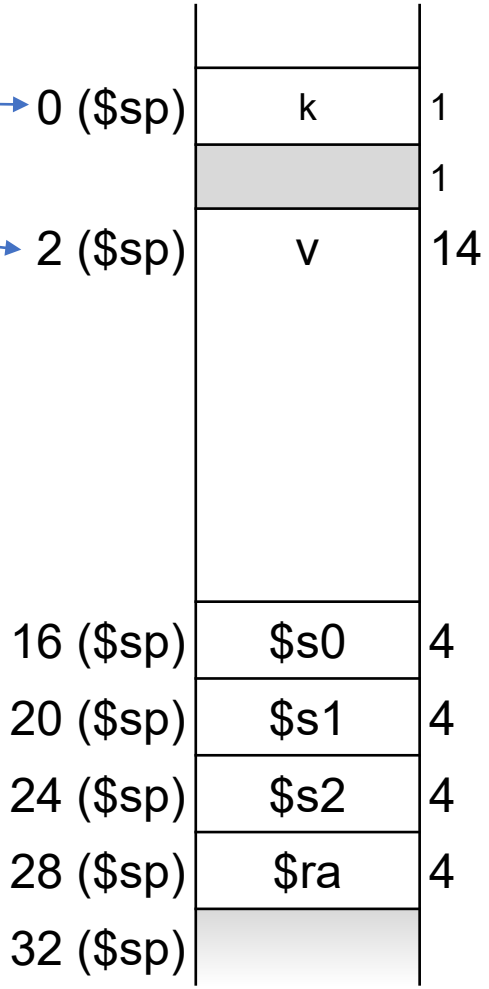
Modificarem registres segurs **$s0, $s1, $s2** (per guardar-hi **a, b, i**), i també **$ra**



Pregunta 4. Subrutines

Dibuixa el bloc d'activació de `subr1` (sobre el diagrama que tens a la dreta), especificant-hi la posició on apunta el registre `$sp` un cop reservat l'espai corresponent a la pila, així com el nom de cada registre i/o variable, i la seva posició (desplaçament relatiu al `$sp`).

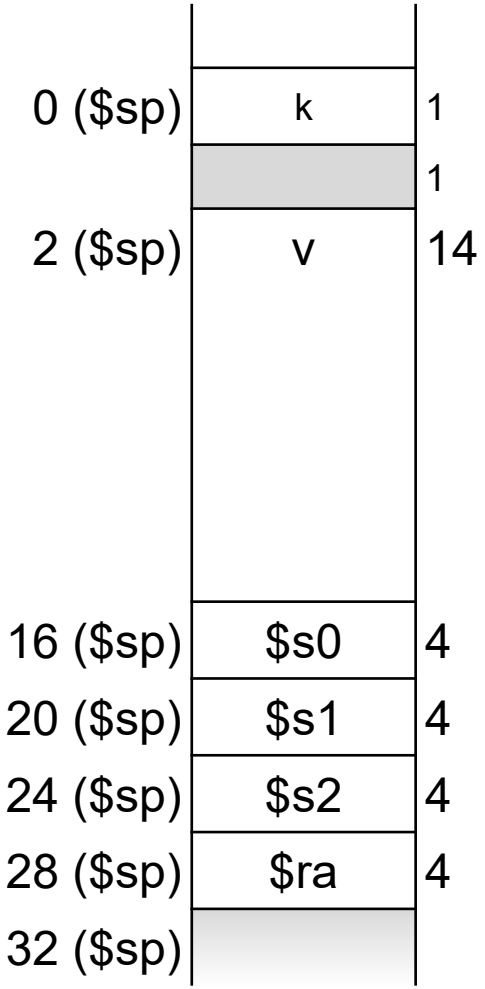
```
void subr1(int a[], int b) {
   char k;
   short v[7];
   int i;
          for (i = 0; i < b; i++)
                 subr2(v, &a[3], &k);
}
```



Pregunta 4. Subrutines

```
subr1:
    addiu   $sp, $sp, -32
    sw      $s0, 16($sp)
    sw      $s1, 20($sp)
    sw      $s2, 24($sp)
    sw      $ra, 28($sp)
    move    $s0, $a0        # a
    move    $s1, $a1        # b
```

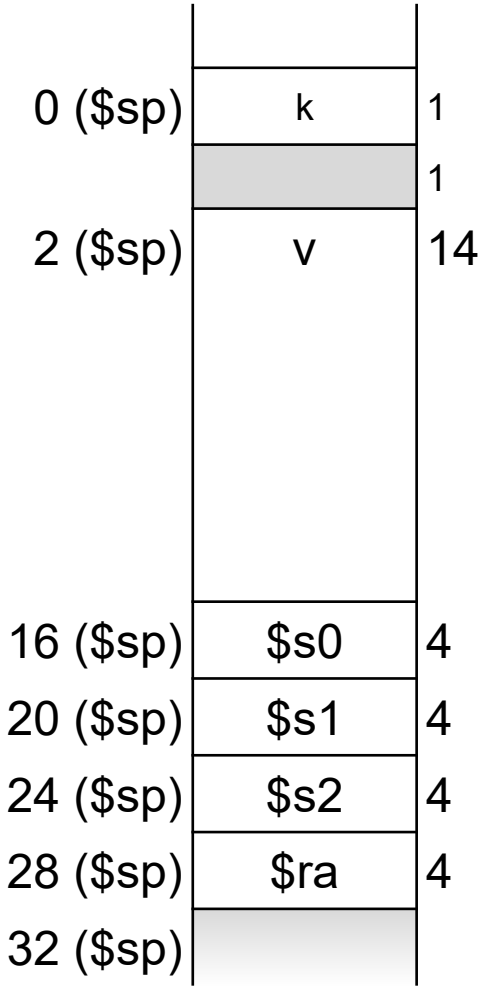
Tradueix a MIPS la subrutina `subr1`.



Pregunta 4. Subrutines

```
subr1:
    addiu   $sp, $sp, -32
    sw      $s0, 16($sp)
    sw      $s1, 20($sp)
    sw      $s2, 24($sp)
    sw      $ra, 28($sp)
    move    $s0, $a0          # a
    move    $s1, $a1          # b
    move    $s2, $zero        # i=0
for:
    bge     $s2, $s1,fifor    # i<b
```

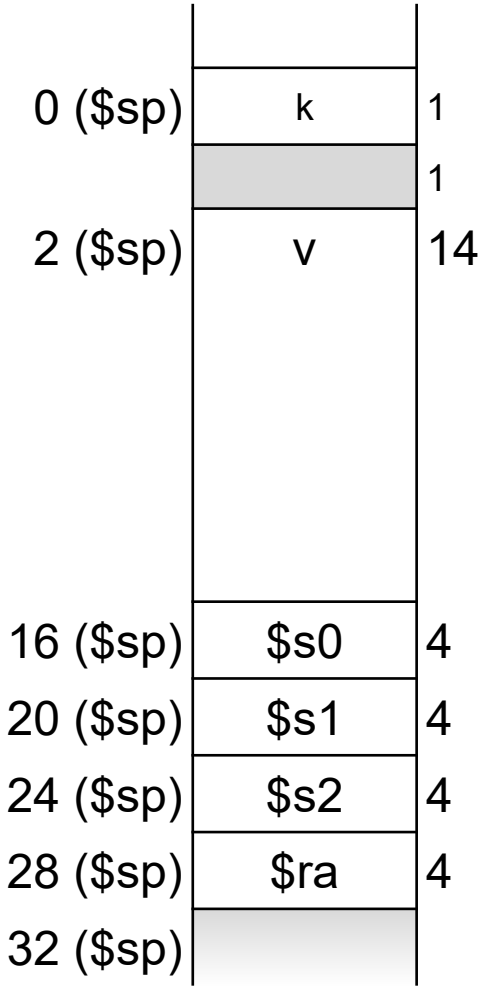
```
for (i = 0; i < b; i++)
        subr2(v, &a[3], &k);
```



Pregunta 4. Subrutines

```
subr1:
    addiu   $sp, $sp, -32
    sw      $s0, 16($sp)
    sw      $s1, 20($sp)
    sw      $s2, 24($sp)
    sw      $ra, 28($sp)
    move    $s0, $a0        # a
    move    $s1, $a1        # b
    move    $s2, $zero      # i=0
for:
    bge     $s2, $s1,fifor  # i<b
    addiu   $a0, $sp, 2     # v
    addiu   $a1, $s0, 12    # &a[3]
    move    $a2, $sp        # &k
```

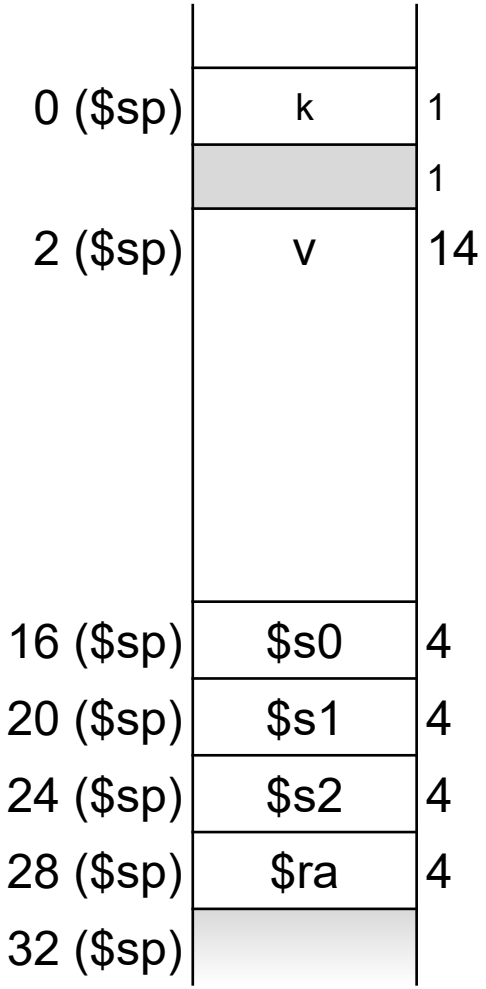
```
for (i = 0; i < b; i++)
        subr2(v, &a[3], &k);
```



Pregunta 4. Subrutines

```
subr1:
    addiu   $sp, $sp, -32
    sw      $s0, 16($sp)
    sw      $s1, 20($sp)
    sw      $s2, 24($sp)
    sw      $ra, 28($sp)
    move    $s0, $a0          # a
    move    $s1, $a1          # b
    move    $s2, $zero        # i=0
for:
    bge     $s2, $s1,fifor    # i<b
    addiu   $a0, $sp, 2       # v
    addiu   $a1, $s0, 12      # &a[3]
    move    $a2, $sp          # &k
    jal     subr2
    addiu   $s2, $s2, 1       # i++
    b       for
fifor:
```

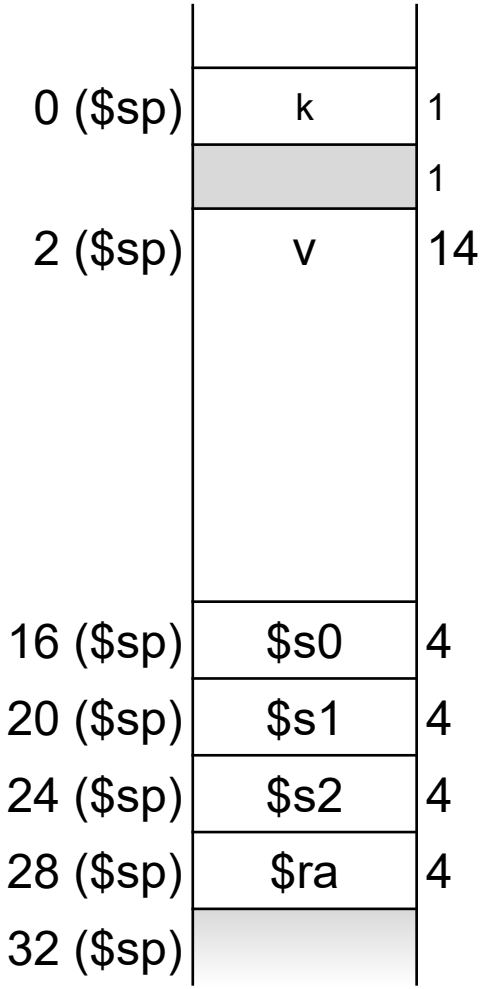
```
for (i = 0; i < b; i++)
        subr2(v, &a[3], &k);
```



Pregunta 4. Subrutines

```
subr1:
    addiu   $sp, $sp, -32
    sw      $s0, 16($sp)
    sw      $s1, 20($sp)
    sw      $s2, 24($sp)
    sw      $ra, 28($sp)
    move    $s0, $a0          # a
    move    $s1, $a1          # b
    move    $s2, $zero        # i=0
for:
    bge     $s2, $s1,fifor    # i<b
    addiu   $a0, $sp, 2       # v
    addiu   $a1, $s0, 12      # &a[3]
    move    $a2, $sp          # &k
    jal     subr2
    addiu   $s2, $s2, 1       # i++
    b       for
fifor:
    lw      $s0, 16($sp)
    lw      $s1, 20($sp)
    lw      $s2, 24($sp)
    lw      $ra, 28($sp)
    addiu   $sp, $sp, 32
    jr      $ra
```

```
for (i = 0; i < b; i++)
        subr2(v, &a[3], &k);
```



Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

```
$f4    = 0100 0000 1000 0000 0000 0000 0000 0011
```

Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

`$f4` = 0100 0000 1000 0000 0000 0000 0000 0011

exponent = 100 0000 1 = 129 − 127 = +2

Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

`$f4` = 0100 0000 1000 0000 0000 0000 0000 0011

exponent = 100 0000 1 = 129 − 127 = +2

`$f4` = **1,**000 0000 0000 0000 0000 0011 × 2^{+2}

Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

`$f4` = 0100 0000 1000 0000 0000 0000 0000 0011

exponent = 100 0000 1 = 129 − 127 = +2

`$f4` = **1,**000 0000 0000 0000 0000 0011 × 2^{+2}

`$f6` = 1011 1111 0111 0000 0000 0000 0000 0101

Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

`$f4` = 0100 0000 1000 0000 0000 0000 0000 0011

exponent = 100 0000 1 = 129 − 127 = +2

`$f4` = **1,**000 0000 0000 0000 0000 0011 × 2^{+2}

`$f6` = 1011 1111 0111 0000 0000 0000 0000 0101

exponent = 011 1111 0 = 126 − 127 = −1

Pregunta 5. Coma flotant

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?

`$f4` = 0100 0000 1000 0000 0000 0000 0000 0011

exponent = 100 0000 1 = 129 − 127 = +2

`$f4` = **1,**000 0000 0000 0000 0000 0011 × 2^{+2}

`$f6` = 1011 1111 0111 0000 0000 0000 0000 0101

exponent = 011 1111 0 = 126 − 127 = −1

`$f6` = **1,**111 0000 0000 0000 0000 0101 × 2^{-1}

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)

$f4: **1,**000 0000 0000 0000 0000 0011 × 2^{+2}

$f6: - **1,**111 0000 0000 0000 0000 0101 × 2^{-1}

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta

```
                                                        G R S
$f4:    1,000 0000 0000 0000 0000 0011                  0 0 0   × 2^+2
$f6: - 0,001 1110 0000 0000 0000 0000                   1 0 1   × 2^+2
```

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

							G	R	S	
$f4:	**1,**000	0000	0000	0000	0000	0011	0	0	0	$\times 2^{+2}$
$f6:	- 0,00**1**	1110	0000	0000	0000	0000	1	0	1	$\times 2^{+2}$
	= 0,110	0010	0000	0000	0000	0010	0	1	1	$\times 2^{+2}$

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

```
                                                        G R S
$f4:    1,000 0000 0000 0000 0000 0011 0 0 0    × 2^+2
$f6:  - 0,001 1110 0000 0000 0000 0000 1 0 1    × 2^+2
      ------------------------------------------------------
      = 0,110 0010 0000 0000 0000 0010 0 1 1    × 2^+2
```

Resultat després de re-normalitzar (si cal):

```
        1,100 0100 0000 0000 0000 0100 1 1 0    × 2^+1
```

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

```
                                                        G R S
$f4:    1,000 0000 0000 0000 0000 0011 0 0 0    × 2^+2
$f6:  - 0,001 1110 0000 0000 0000 0000 1 0 1    × 2^+2
      ------------------------------------------------------
      = 0,110 0010 0000 0000 0000 0010 0 1 1    × 2^+2
```

Resultat després de re-normalitzar (si cal):

```
        1,100 0100 0000 0000 0000 0100 1 1 0    × 2^+1
```

Resultat després d'arrodonir:

```
        1,100 0100 0000 0000 0000 0101          × 2^+1
```

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

							G	R	S	
$f4:	**1,**000	0000	0000	0000	0000	0011	0	0	0	$\times 2^{+2}$
$f6:	- 0,00**1**	1110	0000	0000	0000	0000	1	0	1	$\times 2^{+2}$
	= 0,110	0010	0000	0000	0000	0010	0	1	1	$\times 2^{+2}$

Resultat després de re-normalitzar (si cal):

1,100 0100 0000 0000 0000 0100 1 1 0 $\times 2^{+1}$

Resultat després d'arrodonir:

1,100 0100 0000 0000 0000 0101 $\times 2^{+1}$

Resultat (signe = 0; exponent = +1+127 = 128 = 1000 0000):

0 1000 0000

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

							G	R	S	
$f4:	**1,**000	0000	0000	0000	0000	0011	0	0	0	$\times 2^{+2}$
$f6:	- 0,00**1**	1110	0000	0000	0000	0000	1	0	1	$\times 2^{+2}$
	= 0,110	0010	0000	0000	0000	0010	0	1	1	$\times 2^{+2}$

Resultat després de re-normalitzar (si cal):

1,100 0100 0000 0000 0000 0100 1 1 0 $\times 2^{+1}$

Resultat després d'arrodonir:

1,100 0100 0000 0000 0000 0101 $\times 2^{+1}$

Resultat (signe = 0; exponent = +1+127 = 128 = 1000 0000):

0 1000 0000 100 0100 0000 0000 0000 0101

Pregunta 5. Coma flotant

Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:

- $f4, $f6 tenen signes diferents: cal fer una RESTA
- $f4 té major valor absolut → restarem $f4 menys $f6, i el signe serà el de $f4 (+)
- Igualem l'exponent de $f6 a +2, desplaçant els bits 3 posicions a la dreta
- Calculem la diferència

							G	R	S	
$f4:	**1,**000	0000	0000	0000	0000	0011	0	0	0	$\times 2^{+2}$
$f6:	- 0,00**1**	1110	0000	0000	0000	0000	1	0	1	$\times 2^{+2}$
	= 0,110	0010	0000	0000	0000	0010	0	1	1	$\times 2^{+2}$

Resultat després de re-normalitzar (si cal):

1,100 0100 0000 0000 0000 0100 1 1 0 $\times 2^{+1}$

Resultat després d'arrodonir:

1,100 0100 0000 0000 0000 0101 $\times 2^{+1}$

Resultat (signe = 0; exponent = +1+127 = 128 = 1000 0000):

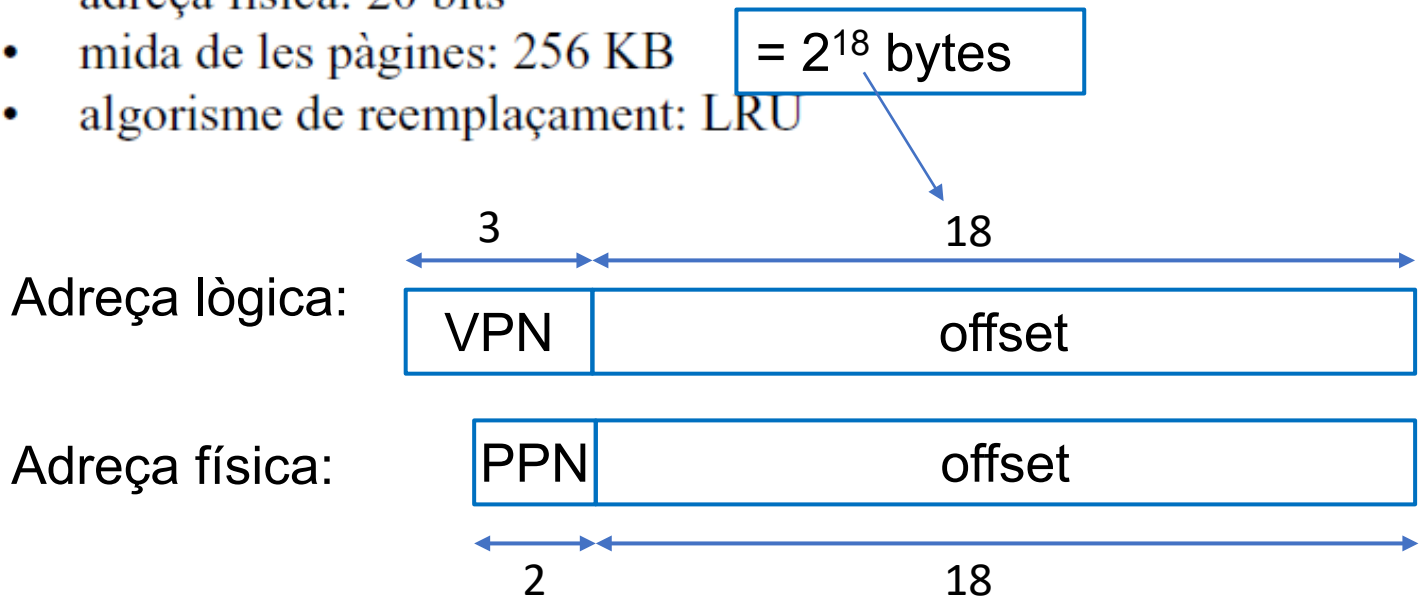
0 1000 0000 100 0100 0000 0000 0000 0101

$f0 = 0x4044 0005

Pregunta 6. Memòria Virtual

Un computador té una memòria virtual paginada amb les següents característiques:

- adreça lògica: 21 bits
- adreça física: 20 bits
- mida de les pàgines: 256 KB
- algorisme de reemplaçament: LRU



Pregunta 6. Memòria Virtual

S'ha començat a executar un programa, i en un moment donat el contingut de la taula de pàgines i de la memòria és el següent:

Contingut de la memòria:

PPN	VPN
0	2
1	3
2	4
3	5

Taula de pàgines:

VPN	PPN	P	D
0	-	0	0
1	-	0	0
2	0	1	1
3	1	1	0
4	2	1	0
5	3	1	0
6	-	0	0
7	-	0	0

LRU: Els 3 darrers accessos han estat a les VPN = 2, 3, 4, 5

Pregunta 6. Memòria Virtual

- Omplir la taula següent. Hi afegim 2 columnes:
  - VPNs presents en MP, per ordre d'accés: pila LRU (modificades en vermell)
  - VPNs presents en MP, per ordre de posició física

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E								
183624	E								
1168AA	L								
15B550	E								
0A8800	L								
077704	L								

Pregunta 6. Memòria Virtual

- Identifiquem les VPNs: els 3 bits de més pes (de 21 bits)
- Per ex. 0x1F6592 = **1 11**11 0110 0101 1001 0010 → VPN = 7

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7							
183624	E	6							
1168AA	L	4							
15B550	E	5							
0A8800	L	2							
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=7. Fallada
- Reemplaça la 2 al marc PPN=0, modificada → escriptura al disc

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN)	MP (VPN)
								~~2~~,3,4,5	~~2~~,3,4,5
1F6592	E	7	s	s	s	2	0	3,4,5,7	7,3,4,5
183624	E	6							
1168AA	L	4							
15B550	E	5							
0A8800	L	2							
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=6. Fallada
- Reemplaça la 3 al marc PPN=1

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7	s	s	s	2	0	~~3~~,4,5,7	7,~~3~~,4,5
183624	E	6	s	s	n	3	1	4,5,7,6	7,6,4,5
1168AA	L	4							
15B550	E	5							
0A8800	L	2							
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=4. Encert (PPN=2)

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7	s	s	s	2	0	3,4,5,7	7,3,4,5
183624	E	6	s	s	n	3	1	4,5,7,6	7,6,4,5
1168AA	L	4	n	n	n		2	5,7,6,4	7,6,4,5
15B550	E	5							
0A8800	L	2							
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=5. Encert (PPN=3). La pàgina queda modificada (D=1)

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7	s	s	s	2	0	3,4,5,7	7,3,4,5
183624	E	6	s	s	n	3	1	4,5,7,6	7,6,4,5
1168AA	L	4	n	n	n		2	5,7,6,4	7,6,4,5
15B550	E	5	n	n	n		3	7,6,4,5	7,6,4,5
0A8800	L	2							
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=2. Fallada
- Reemplaça la 7 al marc PPN=0, modificada → escriptura al disc

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7	s	s	s	2	0	3,4,5,7	7,3,4,5
183624	E	6	s	s	n	3	1	4,5,7,6	7,6,4,5
1168AA	L	4	n	n	n		2	5,7,6,4	7,6,4,5
15B550	E	5	n	n	n		3	~~7~~,6,4,5	~~7~~,6,4,5
0A8800	L	2	s	s	s	7	0	6,4,5,2	2, 6,4,5
077704	L	1							

Pregunta 6. Memòria Virtual

- VPN=1. Fallada
- Reemplaça la 6 al marc PPN=1, modificada → escriptura al disc

@lògica (hex)	L/E	VPN (dec)	fallada de pàg? (s/n)	lectura disc? (s/n)	escript disc? (s/n)	VPN pàg. reempla-çada (dec)	PPN (dec)	LRU (VPN) 2,3,4,5	MP (VPN) 2,3,4,5
1F6592	E	7	s	s	s	2	0	3,4,5,7	7,3,4,5
183624	E	6	s	s	n	3	1	4,5,7,6	7,6,4,5
1168AA	L	4	n	n	n		2	5,7,6,4	7,6,4,5
15B550	E	5	n	n	n		3	7,6,4,5	7,6,4,5
0A8800	L	2	s	s	s	7	0	~~6~~,4,5,2	2, ~~6~~,4,5
077704	L	1	s	s	s	6	1	4,5,2,1	2,1,4,5

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.		
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.		
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		**X**
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.		
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.		
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		**X**
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.	**X**	
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.		
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		**X**
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.	**X**	
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		**X**
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.		
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		**X**
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.	**X**	
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		**X**
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.	**X**	
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
1	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		**X**
2	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.	**X**	
3	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		**X**
4	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.	**X**	
5	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		**X**

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.		
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		
9	Un processador sense TLB pot suportar memòria virtual.		
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.	**X**	
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		
9	Un processador sense TLB pot suportar memòria virtual.		
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.	**X**	
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		**X**
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		
9	Un processador sense TLB pot suportar memòria virtual.		
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.	**X**	
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		**X**
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		**X**
9	Un processador sense TLB pot suportar memòria virtual.		
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.	**X**	
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		**X**
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		**X**
9	Un processador sense TLB pot suportar memòria virtual.	**X**	
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 7. Test

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS

	Afirmació	V	F
6	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.	**X**	
7	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		**X**
8	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		**X**
9	Un processador sense TLB pot suportar memòria virtual.	**X**	
10	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		**X**

Pregunta 8. Matrius

Donat el següent codi en alt nivell:

```
int M[10][10];

main() {
int i,j;
        j=4;
        for (i=0; i<5; i++) {
                M[i][j] = M[4][i+j];
        }
}
```

Completa el següent codi en assemblador del MIPS, que fa el mateix que el programa donat, però que utilitza la tècnica d'accés seqüencial, on `$t0` és el punter als elements `M[i][j]`, `$t1` és el punter als elements `M[4][i+j]` i `$t3` conté l'adreça de la matriu tal que no hi ha d'arribar el punter `$t0` i, per tant, serveix per controlar la finalització del bucle.

Pregunta 8. Matrius

```
j=4;
for (i=0; i<5; i++) {
        M[i][j] = M[4][i+j];
}
```

```
main:
    la      $t0,  M + 16          # punter a M[i][j]
    la      $t1,                  # punter a M[4][i+j]
    la      $t3,                  # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,
    addiu $t1,$t1,
    b bucle
fibucle:
    jr $ra
```

@M[0][4] = M + 4*4

Pregunta 8. Matrius

```
j=4;
for (i=0; i<5; i++) {
        M[i][j] = M[4][i+j];
}
```

```
main:
    la      $t0,  M + 16          # punter a M[i][j]
    la      $t1,  M + 176         # punter a M[4][i+j]
    la      $t3,                  # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,
    addiu $t1,$t1,
    b bucle
fibucle:
    jr $ra
```

@M[4][4] = M + 4*10*4 + 4*4

Pregunta 8. Matrius

```
j=4;
for (i=0; i<5; i++) {
        M[i][j] = M[4][i+j];
}
```

```
main:
    la      $t0,  M + 16          # punter a M[i][j]
    la      $t1,  M + 176         # punter a M[4][i+j]
    la      $t3,  M + 216         # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,
    addiu $t1,$t1,
    b bucle
fibucle:
    jr $ra
```

@M[5][4] = M + 5*10*4 + 4*4

Pregunta 8. Matrius

```
j=4;
for (i=0; i<5; i++) {
        M[i][j] = M[4][i+j];
}
```

```
main:
    la      $t0,    M + 16          # punter a M[i][j]
    la      $t1,    M + 176         # punter a M[4][i+j]
    la      $t3,    M + 216         # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,  40
    addiu $t1,$t1,
    b bucle
fibucle:
    jr $ra
```

stride del punter $t0:
@M[i+1][4] - @M[i][4]

Avança 1 fila = 10 elements = 10*4

Pregunta 8. Matrius

```
j=4;
for (i=0; i<5; i++) {
        M[i][j] = M[4][i+j];
}
```

```
main:
    la      $t0,  M + 16          # punter a M[i][j]
    la      $t1,  M + 176         # punter a M[4][i+j]
    la      $t3,  M + 216         # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,  40
    addiu $t1,$t1,  4
    b bucle
fibucle:
    jr $ra
```

stride del punter $t1:
@M[4][(i+1)+4] - @M[4][i+4]

Avança 1 element = 4 bytes

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	
0x10010001	
0x10010002	
0x10010003	
0x10010004	
0x10010005	
0x10010006	
0x10010007	
0x10010008	
0x10010009	
0x1001000A	
0x1001000B	
0x1001000C	
0x1001000D	
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	
0x10010009	
0x1001000A	
0x1001000B	
0x1001000C	
0x1001000D	
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

Pregunta 9. Assemblador

```
v1:.word 0x01001
v2:.half 5,-2, (
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	
0x10010029	
0x1001002A	
0x1001002B	
0x1001002C	
0x1001002D	
0x1001002E	
0x1001002F	

Pregunta 9. Assemblador

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

@Memòria	Dada
0x10010030	00
0x10010031	00
0x10010032	00
0x10010033	00
0x10010034	00
0x10010035	00
0x10010036	00
0x10010037	00
0x10010038	00
0x10010039	00
0x1001003A	00
0x1001003B	00
0x1001003C	00
0x1001003D	00
0x1001003E	00
0x1001003F	00

Pregunta 9. Assemblador

b) Quin és el valor final del registre `$t2`, en hexadecimal, després d'executar el següent fragment de codi?

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

Pregunta 9. Assemblador

b) Quin és el valor final del registre `$t2`, en hexadecimal, després d'executar el següent fragment de codi?

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

Pregunta 9. Assemblador

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

$t1 ← 0x10010010

v1:

+4

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

Pregunta 9. Assemblador

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

$t1 ← 0x10010010

$t2 ← 0x000089AB

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

+8

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

Pregunta 9. Assemblador

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

$t1 ← 0x10010010

$t2 ← 0x000089AB

$t3 ← 0xFFFFFFE8

(extensió de signe)

+16

@Memòria	Dada
0x10010000	0C
0x10010001	10
0x10010002	00
0x10010003	01
0x10010004	10
0x10010005	00
0x10010006	01
0x10010007	10
0x10010008	05
0x10010009	00
0x1001000A	FE
0x1001000B	FF
0x1001000C	34
0x1001000D	12
0x1001000E	01
0x1001000F	00

@Memòria	Dada
0x10010010	11
0x10010011	11
0x10010012	00
0x10010013	00
0x10010014	00
0x10010015	00
0x10010016	00
0x10010017	00
0x10010018	AB
0x10010019	89
0x1001001A	00
0x1001001B	00
0x1001001C	00
0x1001001D	00
0x1001001E	00
0x1001001F	00

@Memòria	Dada
0x10010020	E8
0x10010021	0A
0x10010022	00
0x10010023	00
0x10010024	00
0x10010025	00
0x10010026	00
0x10010027	00
0x10010028	00
0x10010029	00
0x1001002A	00
0x1001002B	00
0x1001002C	00
0x1001002D	00
0x1001002E	00
0x1001002F	00

Pregunta 9. Assemblador

```
la      $t0, v1+4
lw      $t1, 0($t0)
lw      $t2, 8($t1)
lb      $t3, 16($t1)
xor     $t2, $t3, $t2
```

$t1 ← 0x10010010

$t2 ← 0x000089AB

$t3 ← 0xFFFFFFE8

(extensió de signe)

$t3 =	1111	1111	1111	1111	1111	1111	1110	1000
xor $t2 =	0000	0000	0000	0000	1000	1001	1010	1011
=	1111	1111	1111	1111	0111	0110	0100	0011

$t2 ← **0xFFFF7643**

FI

Gràcies a tots

Ha estat un plaer estar amb vosaltres

Espero que tingueu un bon aprofitament del curs

Joan Manuel Parcerisa