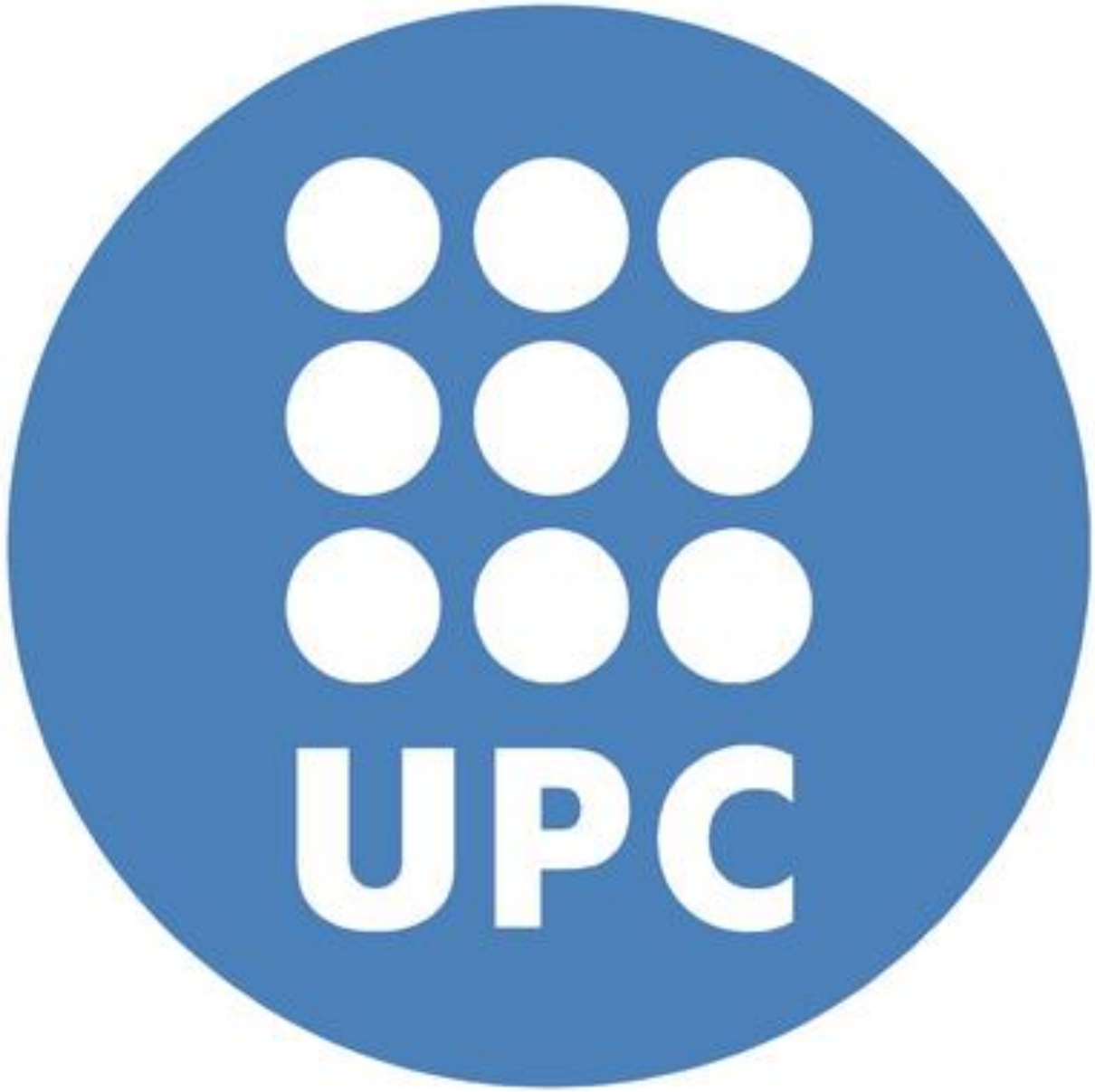
ESC2: PRÀCTICA LABORATORI 3



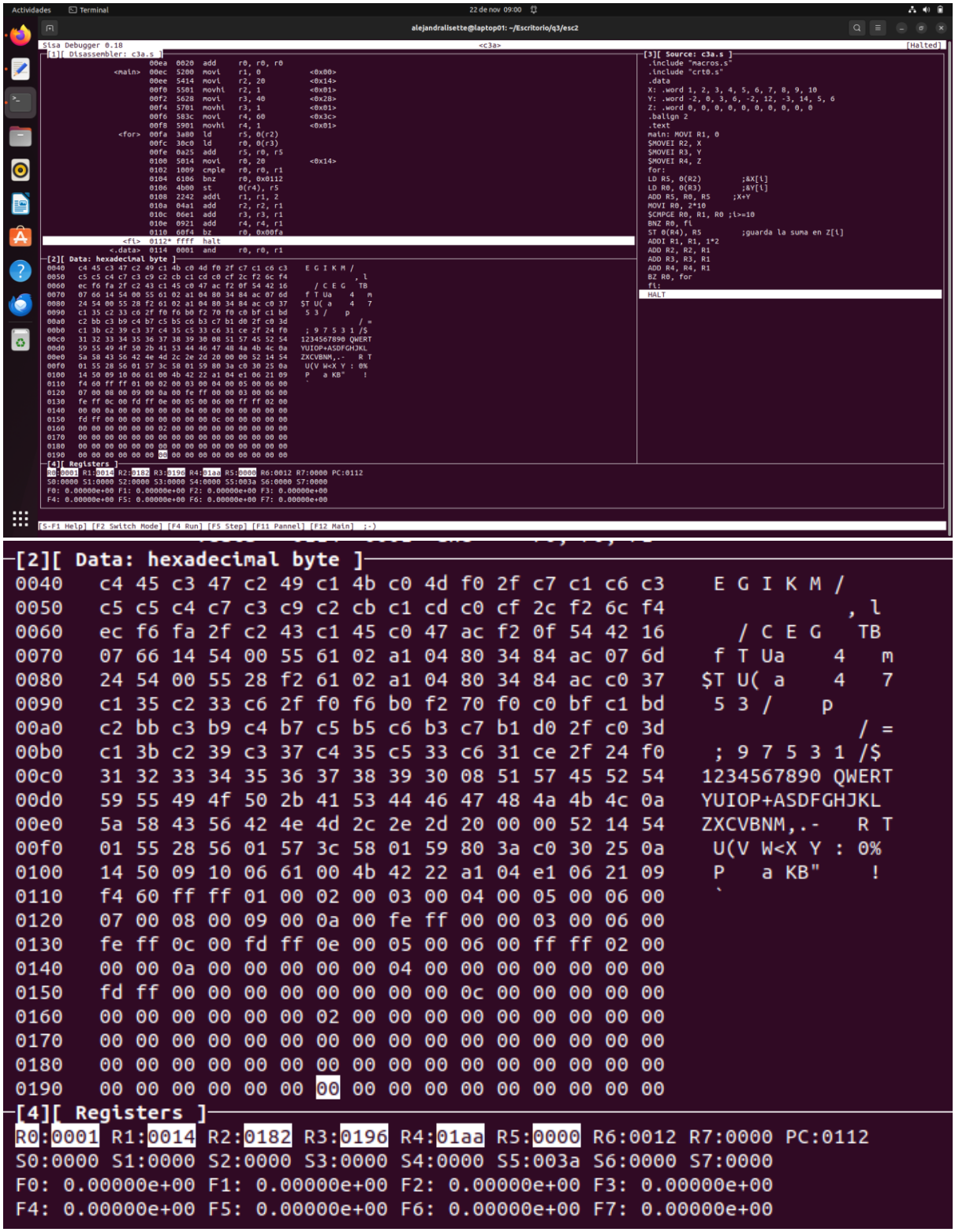
Nom de la professora: Ana M. Heredero Lazaro
Nom de l'alumne: Alejandra Lisette Rocha (Grup I3021)

Activitat 3.A:

Exercici 3.1: Tradueix el codi anterior a SISA-F.

```
main(){
        int i;
        int X[]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int Y[]={-2, 0, 3, 6, -2, 12, -3, 14, 5, 6};
        int Z[10];
        for(i=0; i<10;i++)
        {
                Z[i]=X[i]+Y[i];
        }
}
```

```
.include "macros.s"
.include "crt0.s"
.data
X: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Y: .word -2, 0, 3, 6, -2, 12, -3, 14, 5, 6
Z: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .balign 2
        .text
main:
MOVI R1, 0
$MOVEI R2, X
$MOVEI R3, Y
$MOVEI R4, Z
for:
        LD R5, 0(R2)            ;R5=&X[i]
        LD R0, 0(R3)            ;R0=&Y[i]
        ADD R5, R0, R5          ;R5=X+Y
        MOVI R0, 2*10           ;R0 conté el 10
        $CMPGE R0, R1, R0 ; si i>=10 llavors R0=1
        BNZ R0, fi              ;si R0=1 llavors salta a la etiqueta fi
        ST 0(R4), R5            ;guarda la suma en Z[i]
        ADDI R1, R1, 1*2        ;R1=++i;
        ADD R2, R2, R1          ;R2=X[i];
        ADD R3, R3, R1          ;R3=Y[i];
        ADD R4, R4, R1          R4=Z[i];
        BZ R0, for              ;si R0=0 salta a la etiqueta for
        fi:
                HALT
```



Exercici 3.2:
Quant els tinguis traudit comprova en el simulador les adreces assignades als vectors X, Y i Z. Amb l'adreça inicial de X, Y, Z dedueix les adreces de:

@X[0]=0114	**@Y[0]=0128**	**@Z[0]=013C**
@X[1]=0116	**@Y[1]=012A**	**@Z[1]=013E**

@X[2]=0118	@Y[2]=012C	@Z[2]=0140
@X[3]=011A	@Y[3]=012E	@Z[3]=0142
@X[4]=011C	@Y[4]=0130	@Z[4]=0144
@X[5]=011E	@Y[5]=0132	@Z[5]=0146
@X[6]=0120	@Y[6]=0134	@Z[6]=0148
@X[7]=0122	@Y[7]=0136	@Z[7]=014A
@X[8]=0124	@Y[8]=0138	@Z[8]=014C
@X[9]=0126	@Y[9]=013A	@Z[9]=014E

Repàs: Quants bits tenen les adreces en SISA-F?

Tenen 16 bits

Activitat 3.B: Volem que el modificar el codi anterior per tal de que ens mostri per pantalla la direcció de línea de cache en cadascún dels accesos a cache, tant de X, Y i Z. Obtindrem també el tag (etiqueta) i l'observarem al simulador encara que no el mostrem per pantalla. Suposarem que la cache que tindria el processador és una cache directa de 512 línies, i on cada línea són 4 paraules (Recorda que a més a més 1 paraula son 2 bytes en SISA-F). Tenint en compte aquestes dades, completa el següent pseudo-codi en alt nivell:

```
main() {
int i, line, line_cache, tag, address, j;
char character;
int fil=4, col;
int X[]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int Y[]={-2, 0, 3, 6, -2, 12, -3, 14, 5, 6};
int Z[10];
for(i=0; i<10;i++) {
        Z[i]=X[i]+Y[i];
        col=15;
/* afegeix una o més instruccions que obtinguin l'adreça de X[i] i la posi a la variable
address */
Adress = &x+2*i;
line= adress>>3; /* obtenim la línea o bloc de memoria principal */ (para sacarnos el
offset)
line_cache= line & 0x01FF; /*completa posant la màscara adecuada per obtenir */
/* el número de línea a memoria cache */
tag= adress>>12; /*completa per obtenir el tag */
/*Ara treurem el valor de line_cache en binari per pantalla, per això ho haurem de
convertir a ASCII, bit a bit */ (los caracteres qeu salen tienen que ser los de ascii )
out(Rfil_pant, fil);
while(line_cache!=0) {
        caracter= line_cache&1; /* obtenim el bit de menys pes */
        caracter=caracter +48; /*completa que hem de sumar per transformar-lo en
        ASCII*/
        out(Rcol_pantalla, col);
        out(Rdat_pant, character);
        out(Rcont_pantalla, 0x8000);
        line_cache=line_cache>>1; /*completa per desplaçar un bit */
```

```
        col--;
}
col=30; /* posem al columna a 30 */
/*repetim el mateix per line_cache de Y[i] */
/* afegeix una o més instruccions que obtinguin l'adreça de Y[i] i la posi a la variable
address */
&y+2*i; ;
col=45; /*posem la columna a 45 */
/*repetim el mateix per line_cache de Z[i] */
/* afegeix una o més instruccions que obtinguin l'adreça de Z[i] i la posi a la variable
address */
&z+2*i;
/* per la següent iteració del bucle en i incrementem la fila */
fil++;
        }
}
```

Exercici 4.1: Tradueix el codi anterior a SISA-F i simula'l pas a pas.
NOTA.Adjunta en un *.s apart el codi traduit.

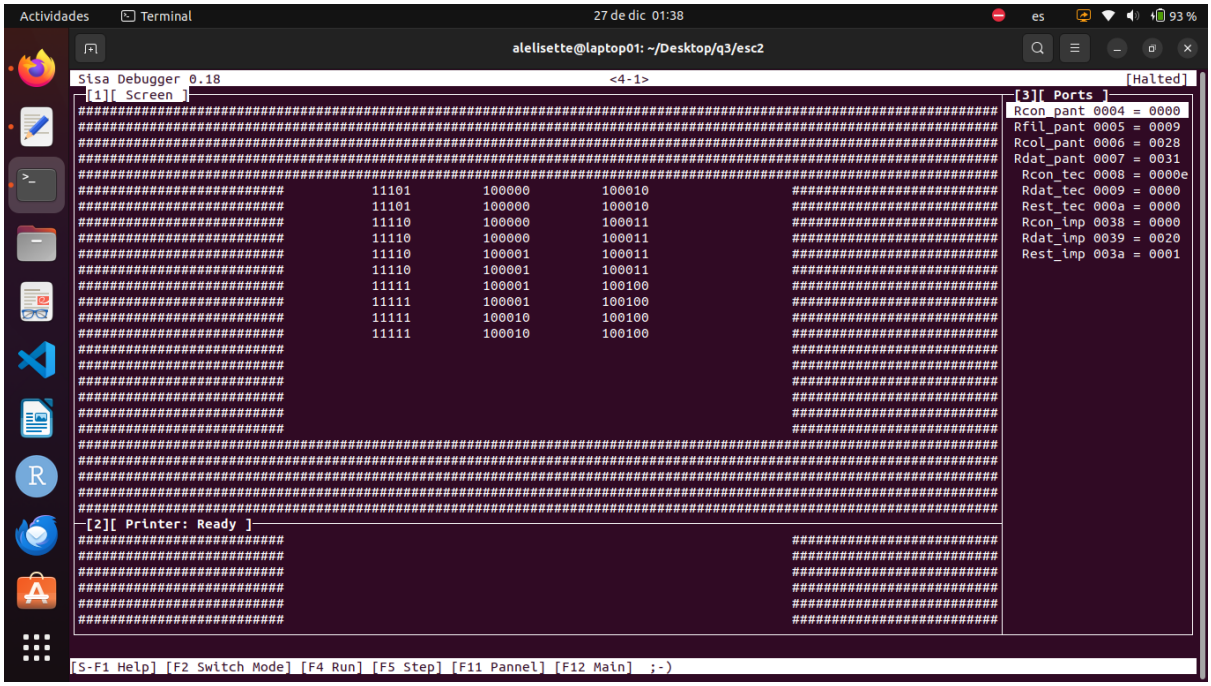
```
.include "macros.s"
.include "crt0.s"

   X: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
   Y: .word -2, 0, 3, 6, -2, 12, -3, 14, 5, 6
   Z: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   col1:    .word 15
   col2:    .word 30
   col3:    .word 45
   fil:    .word 0
   address:  .word 0

   .balign 2
   .text

main:

   MOVI R1, 0    ; R1= bucle for
   $MOVEI R2, X
   $MOVEI R3, Y
   $MOVEI R4, Z

   for:
        LD R5, 0(R2)
        LD R6, 0(R3)
        ADD R6, R6, R5
        ST 0(R4), R6

        ; A continuació mostrarem el line_cache de X per pantalla
```

```
$MOVEI R0, address ;carrguem la adreça adress dins de R0
ST 0(R0), R2    ;guardem dins de R0 el valor de @X[i]

$MOVEI R6, -3    ;R6 conté la line=adress>>3
SHL R5, R2, R6

$MOVEI R6, 0x01FF    ;line_cache=line&0x01FF
AND R5, R5, R6        ;R5 = line_cache

$MOVEI R6, -12 ;R6 = adress>>12
SHL R6, R2, R6    ; en R6 conté el tag

$MOVEI R0, fil          ;R0=fil
LD R0, 0(R0)
OUT Rfil_pant, R0    ;mostra per pantalla el line_cache

while1:
        $MOVEI R0, 1
        AND R6, R5, R0
        $MOVEI R0, 48
        ADD R6, R6, R0
        $MOVEI R0, col1
        LD R0, 0(R0)
        OUT Rcol_pant, R0
        OUT Rdat_pant, R6
        $MOVEI R0, 0x8000
        OUT Rcon_pant, R0
        $MOVEI R0, -1          ;mou el line_cache un bit hacia la dreta
        SHL R5, R5, R0
        $MOVEI R0, col1    ;R0=col1
        LD R0, 0(R0)            ;llegeix cada contingut de posició dins de R0
        ADDI R0, R0, -1    ;R0 decreix de un en un
        $MOVEI R6, col1
        ST 0(R6), R0
        BNZ R5, while1          ;Salta a while1 en cas de que R5 sigui igual a 1

$MOVEI R6, col1
MOVI R0, 15
ST 0(R6), R0    ;col1 val 15

;A continuació, mostrem el line_cache de Y per pantalla

$MOVEI R0, address    ;carrguem la adreça adress dins de R0
ST 0(R0), R3    ;R0 conté @Y[i]

$MOVEI R6, -3    ;adress>>3
SHL R5, R3, R6    ;R5 conté line desplaçat
```

```
$MOVEI R6, 0x01FF    ;R6 conté 0x01FF
AND R5, R5, R6    ; R5=line & 0x01FF

$MOVEI R6, -12    ;adress>>12
SHL R6, R3, R6    ;R6 conté el tag desplaçat

$MOVEI R0, fil
LD R0, 0(R0)
OUT Rfil_pant, R0 ;mostrem per pantalla el line_cache

while2:
        $MOVEI R0, 1
        AND R6, R5, R0
        $MOVEI R0, 48
        ADD R6, R6, R0
        $MOVEI R0, col2
        LD R0, 0(R0)
        OUT Rcol_pant, R0
        OUT Rdat_pant, R6
        $MOVEI R0, 0x8000
        OUT Rcon_pant, R0
        $MOVEI R0, -1                ;mou line_cache un bit hacia la dreta
        SHL R5, R5, R0
        $MOVEI R0, col2      ; col--
        LD R0, 0(R0)
        ADDI R0, R0, -1
        $MOVEI R6, col2
        ST 0(R6), R0
        BNZ R5, while2

$MOVEI R6, col2
MOVI R0, 30
ST 0(R6), R0    ;col2=30

;A continuació, mostrem el line_cache de Z per pantalla

$MOVEI R0, address    ;carreguem adress dins de R0
ST 0(R0), R4            ;Guarda dins de R0 el @Z[i]

$MOVEI R6, -3    ;adress>>3
SHL R5, R4, R6    ; R5 conté line

$MOVEI R6, 0x01FF
AND R5, R5, R6    ; R5 = line & 0x01FF

$MOVEI R6, -12        ;R6=-12
SHL R6, R4, R6    ; R6 conté el tag desplaçat
```

```
        $MOVEI R0, fil          ;R0 conté fil
        LD R0, 0(R0)
        OUT Rfil_pant, R0    ;mostrem line_cache per pantalla

        while3:
                $MOVEI R0, 1
                AND R6, R5, R0
                $MOVEI R0, 48
                ADD R6, R6, R0
                $MOVEI R0, col3
                LD R0, 0(R0)
                OUT Rcol_pant, R0
                OUT Rdat_pant, R6
                $MOVEI R0, 0x8000
                OUT Rcon_pant, R0
                $MOVEI R0, -1           ;mou line_cache un bit hacia la dreta
                SHL R5, R5, R0
                $MOVEI R0, col3       ;R0=@col3
                LD R0, 0(R0)
                ADDI R0, R0, -1    ;col--
                $MOVEI R6, col3
                ST 0(R6), R0
                BNZ R5, while3

        $MOVEI R6, col3
        MOVI R0, 45    ;R0 conté 45
        ST 0(R6), R0    ;col3=45
        $MOVEI R0, fil
        LD R0, 0(R0)
        ADDI R0, R0, 1    ;fil++
        $MOVEI R6, fil
        ST 0(R6), R0
        ADDI R2, R2, 2    ;incrementem la posició del vector X en 2
        ADDI R3, R3, 2    ;incrementem la posició del vector Y en 2
        ADDI R4, R4, 2    ;incrementem la posició del vector Z en 2
        ADDI R1, R1, 1    ;incrementem el iterador del bucle for, i++
        MOVI R0, 10
        $CMPLT R0, R1, R0
        $MOVEI R6, for
        JNZ R0, R6    ;salta a R6 si R0 es 1
    HALT
```

Exercici 4.2. Quants bits té line_cache? Line_cache té 16 bits.
Com escriurà en pantalla? Escriura els bits de cada vector després de la modificació.

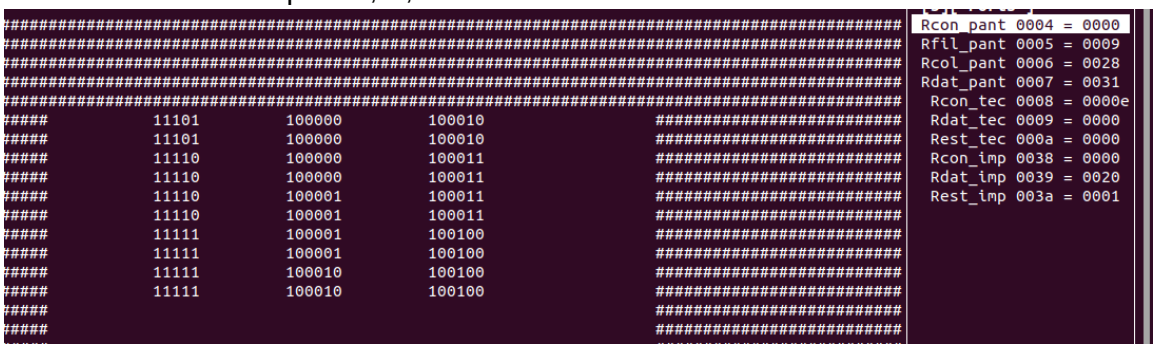


Què escriu en cada fila? cadascuna de les adreços dels vectors X[i],Y[i] i Z[i]

I en cada columna? les adreces X[0]...X[9], Y[0]...Y[9] i Z[0]...fins a Z[9] amb la diferencia de que ara descartarà els 3 bits de menys pes, o sigui el byte offset i el word offset, amb la màscara descarteme el tag i que només es quedin els 9 bits de line_cache.

Si creus que no tindràs suficient lloc per escriure a cada fila pots modificar els paràmetres, fil i col, adecuadament.

Exercici 4.3. Executa el codi anterior i fes una captura de pantalla (adjunta-la) del que has obtingut. Coincideix el resultat de line_cache de cada accés del que obtindries a partir de les adreces de X, Y i Z. IMPORTANT: Al canviar el codi t'hauran canviat les posicions de memòria on es col·loquen X, Y, Z.



Per poder extreure i comparar la línia i el TAG torna a obtenir les direccions inicial de X, Y, Z. Cada quantes dades canvia la linea de cache? Cada 4 words

Exercici 4.4. Observa també amb el simulador el tag que obtens per X, Y i Z.

És el mateix per tots els accesos a X? i a Y? i a Z? Si, és igual i per a totes direccions és 0. Això és perquè des de la primera direcció de memoria del vector X=0x00ec fins la darrera dirección de memoria del vector Z,és a dir, Z[9]=0x0126 els 4 bits més significatius que formen el TAG son 0's.

Apunta aquí el resultat i compara'l amb el que obtindries a partir de les adreces inicials de X, Y i Z.

X[0] = 0x00ec després de descartar els 3 bits de menys es quedarien com 11101

…

X[9] = 0x00fe després de descartar els 3 bits de menys pes es quedarien com 11111

Y[0] = 0x0100 després de descartar els 3 bits de menys pes es quedarien com 100000
…
Y[9] = 0x0112 després de descartar els 3 bits de menys pes es quedarien com 100010

Z[0] = 0x0114 després de descartar els 3 bits de menys pes es quedarien com 100010
…
Z[9] = 0x0126 després de descartar els 3 bits de menys pes es quedarien com 100100