ESTRUCTURA DE COMPUTADORS 2:

LABORATORI 1

Professora: Ana M.Heredero

Alumnes: Alejandra Lisette Rocha

4.1 Escriu un programa en alt nivell que mostri una ‘A’ a la posició [4,8] de la pantalla, amb atribut invers, i una ’B’ a la posició [4,9] amb atribut normal. Observa que la fila és la mateixa en els dos casos, així que sols cal escriure-la un cop.

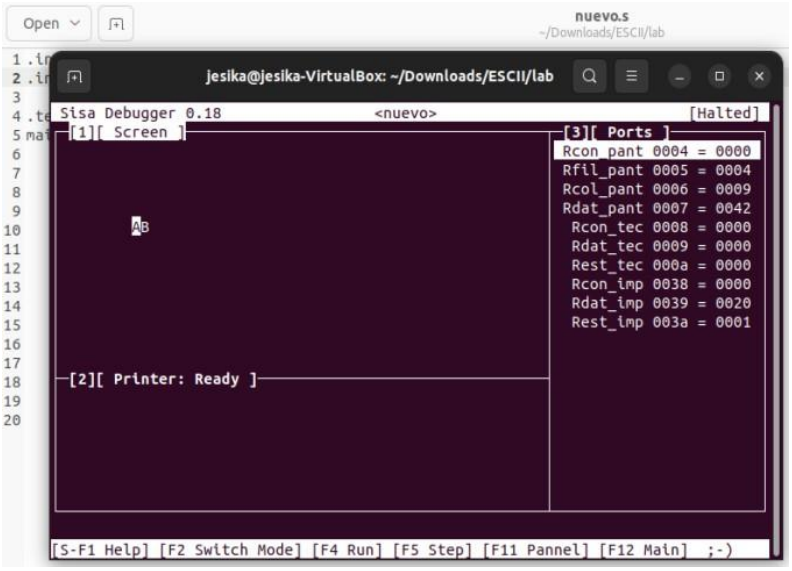
C:

```
include main() {
        int fila = 4;
        int columna = 8;
        int digitocontrol = 0x8000;
        OUT(Rfil_pant, fila);
        OUT(Rcol_pant, columna);
        OUT(Rdat_pant, 0x100 + ‘A’);
        OUT(Rcon_pant, digitocontrol);
        columna++;
        OUT(Rcol_pant, columna);
        OUT(Rdat_pant, ‘B’);
        OUT(Rcon_pant, digitocontrol);
}
```

SISA-F:

```
.include “macros.s”
.include “crt0.s”

.text
main:
        MOVI R1, 4
        MOVI R2, 8
        OUT Rfil_pant, R1
        OUT Rcol_pant, R2
        $MOVEI R3, 0x100+ ‘A
        OUT Rdat_pant, R3
        $MOVEI R4, 0x8000
        OUT Rcon_pant, R4
```

```
        ADDI R2, R2, 1
        OUT Rcol_pant, R2
        $MOVEI R3, ‘B
        OUT Rdat_pant, R3
        OUT Rcon_pant, R4
HALT
```



Exercici 4.2: Escriu un programa en alt nivell que esperi fins que es polsi una tecla qualsevol, i llavors escrigui una ‘A’ a la posició [4,8] de la pantalla, amb atribut normal.

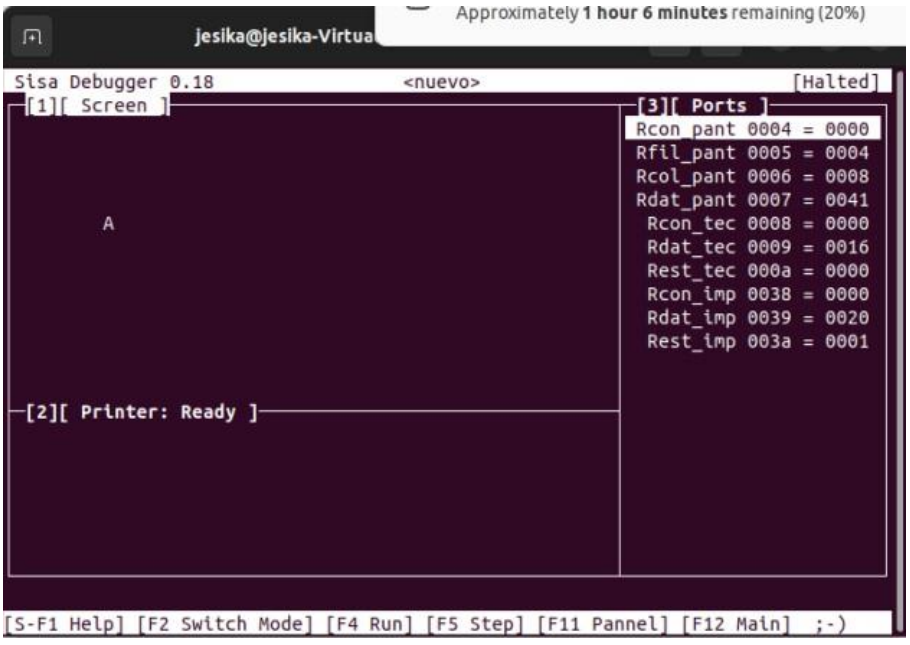
C:

```
int main() {

        int var = in(Rest_tec);
        while(var != 1) {
        var = in(Rest_tec);
        }

        in(Rdat_tec);
        OUT(Rfil_pant, 4);
        OUT(Rcol_pant, 8);
        OUT(Rdat_pant, ‘A’);
```

```
        OUT(Rcon_pant, 0x8000);
}
```

SISA F:

```
do: IN R1, Rest_tec
        BZ R1, do

        IN R1, Rdat_tec
        MOVI R2, 4
        MOVI R3, 8
        OUT Rfil_pant, R2
        OUT Rcol_pant, R3
        $MOVEI R4, ‘A
        OUT Rdat_pant, R4
        $MOVEI R4, 0x8000
        OUT Rcon_pant, R4

        HALT
```



Exercici 4.3: Escriu una nova versió del programa en alt nivell anterior per tal que, en comptes d'una 'A', escrigui el caràcter associat a la tecla polsada. Fixa't que es tracta de fer sols un petit canvi en la finalització. Recorda que per traduir el codi de rastreig del teclat es disposa del vector tteclat. Ara, tradueix l'anterior codi a assemblador SISA-F en el mateix fitxer s4b.s(sols cal afegir les modificacions). Assembla'l i comprova que funciona, amb el simulador.

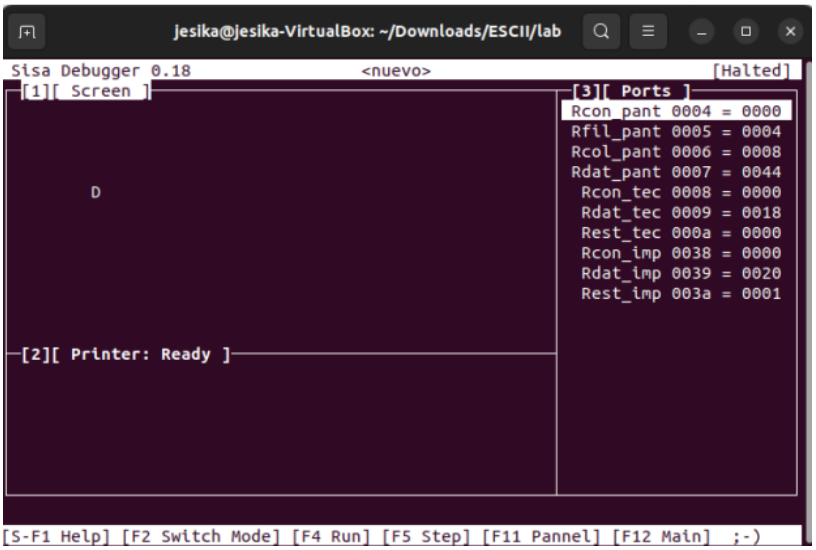
C:

```
int main() {
        int var = in(Rest_tec);
        while(var != 1) {
        var = in(Rest_tec);
        }
        char letra: in(Rdat_tec);
        OUT(Rfil_pant, 4);
        OUT(Rcol_pant, 8);
        OUT(Rdat_pant, letra);
        OUT(Rcon_pant, 0x8000);
}
```

SISA F:

```
do: IN R1, Rest_tec
        BZ R1, do

        IN R1, Rdat_tec
        $MOVEI R4, tteclat
        ADD R1, R4, R1
        LDB R1, 0(R1)

        MOVI R2, 4
        MOVI R3, 8
        OUT Rfil_pant, 4
        OUT Rcol_pant, 8
```

```
OUT Rdat_pant, R1
$MOVEI R4, 0x8000
OUT Rcon_pant, R4

HALT
```



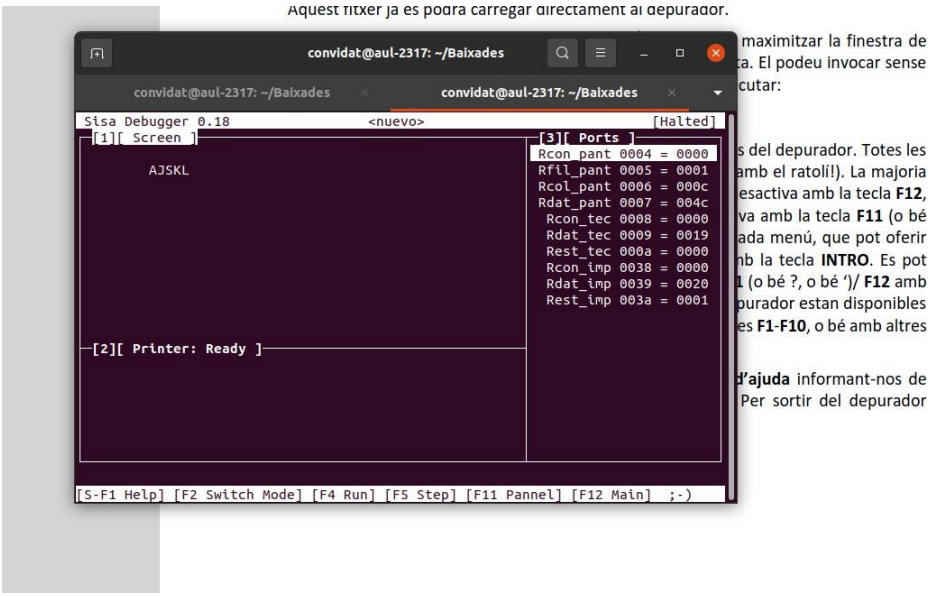
Exercici 4.4: Escriu en alt nivell una nova versió del programa anterior, tal que faci la mateixa tasca (llegir una tecla i escriure-la en pantalla) però repetidament, fins que la tecla polsada sigui una 'F'. Fixa't que es tracta sols d'insertar l'anterior programa dins un bucle.

```
C:
int main() {

int num = in(Rest_tec);
        while(num != 1) {
        num = in(Rest_tec);
        }
        char letra = in(Rdat_tec);
        while(letra != 'F') {
        letra = in(Rdat_tec);
        OUT(Rfil_pant, 4);
        OUT(Rcol_pant, 8);
```

```
        OUT(Rdat_pant, letra);
        OUT(Rcon_pant, 0x8000);
        }
}
```

SISA-F

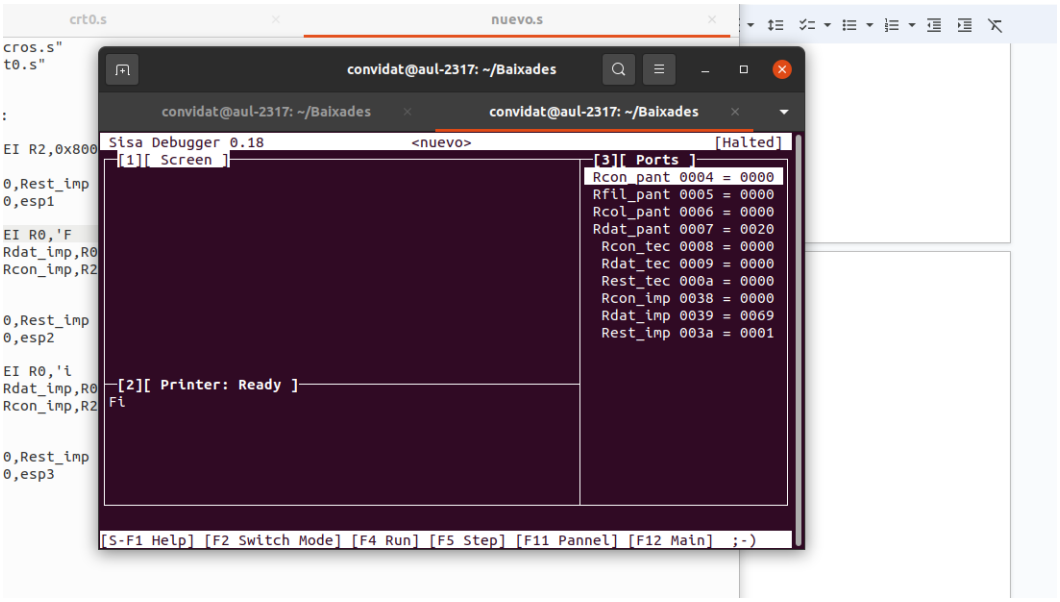
```
        .include "macros.s"
        .include "crt0.s"

        .text
        main:
            MOVI R2, 4
            MOVI R3, 8
        do:
            IN R2, Rest_tec
            BZ R2, do

            IN R1, Rdat_tec
            $MOVEI R4, tteclat
            ADD R1, R4, R1
            LDB R1, 0(R1)
            $MOVEI R5, 'F
            CMPEQ R5, R5, R1
            BNZ R5, fi


            OUT Rfil_pant, R2
            OUT Rcol_pant, R3
            OUT Rdat_pant, R1
            $MOVEI R4, 0x8000
            OUT Rcon_pant, R4
            ADDI R3, R3, 1
            BZ R5, do

        fi: HALT
```



Exercici 4.5: Fes un programa en alt nivell que imprimeixi la paraula "Fi". Abans d'imprimir cada caràcter cal esperar que estigui preparada. I al final del programa, també.

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

.text
        main:

    $MOVEI R2,0x8000
  esp1:
    IN R0,Rest_imp
    BZ R0,esp1

    $MOVEI R0,'F
    OUT Rdat_imp,R0
    OUT Rcon_imp,R2
```

```
esp2:
    IN R0,Rest_imp
    BZ R0,esp2

    $MOVEI R0,'i
    OUT Rdat_imp,R0
    OUT Rcon_imp,R2

esp3:
    IN R0,Rest_imp
    BZ R0,esp3

HALT
```

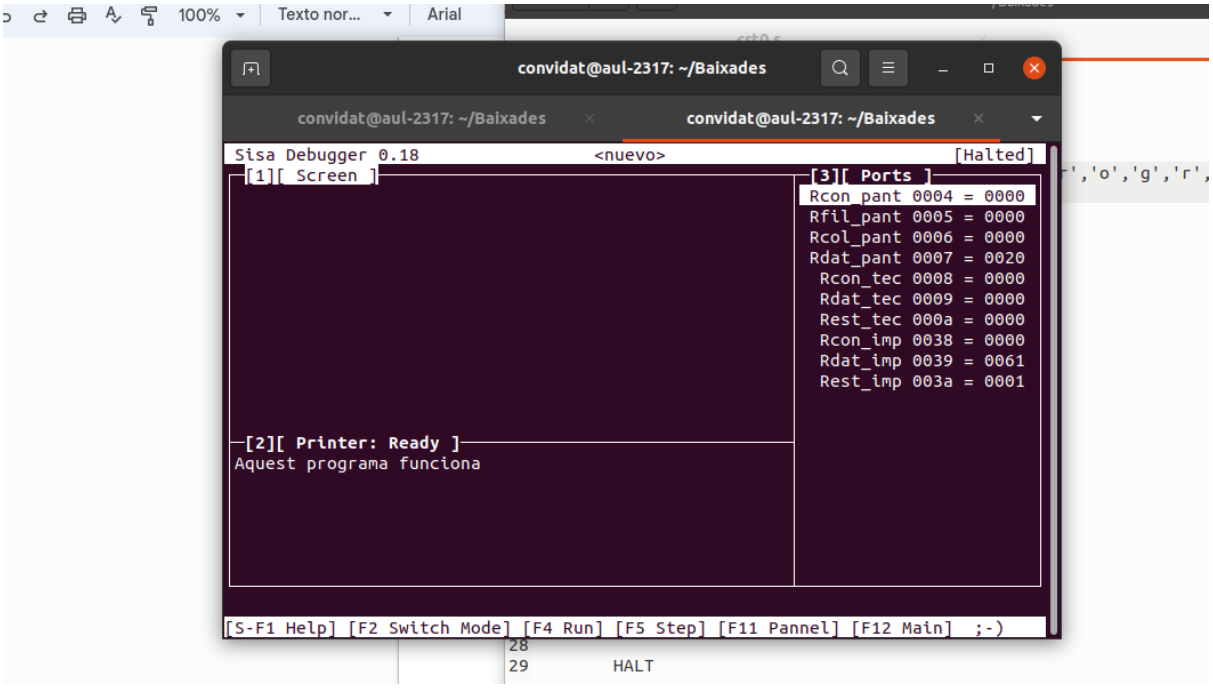


Exercici 4.6:

Escriu en alt nivell una nova versió del programa anterior que escrigui el vector frase a la impressora. Fixa't que el vector és un string, per tant, acaba amb un byte a zero (valor binari 0). Es tracta doncs de fer un bucle que iteri fins a trobar el zero final.

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

```
.include "crt0.s"

.data

        frase: .ascii "Aquest programa funciona"
        .balign 2

.text
    main:
        $MOVEI R1, frase
        $MOVEI R2, 0x8000
   do:
        IN R0, Rest_imp
        BZ R0, do

        LDB R3, 0(R1)
        BZ R3, enndo

        OUT Rdat_imp, R3
        OUT Rcon_imp, R2

        ADDI R1,R1,1
        BNZ R3, do
 enndo:
         IN R0, Rest_imp
         BZ R0, enndo

         HALT
```



ACTIVIDAD 4D

Visualització d'operacions de tractament de bits L'objectiu d'aquest apartat és combinar l'accés a la pantalla, al teclat, i les operacions de tractament de bits, i ho farem en 3 parts: En primer lloc, escriurem un programa que mostri a les posicions [0,0]-[0,15] de la pantalla els 16 bits de la representació en binari (uns i zeros) de la variable w de mida word (figura 4.1), mostrant el bit més significatiu a la columna 0 i el menys significatiu a la columna 15. El programa main fa una crida a la subrutina mostra passant-li com a paràmetre per valor la variable w. Aquesta subrutina no retorna res ja que és una acció, però mostra la representació binària del paràmetre i a les posicions [0,0]-[0,15]. Completa l'exercici 4.7 abans de seguir

```
int w=0x8888;
main(){
mostra(w);
}
void mostra(int i){
register int col=16;
out (Rfil_pant,0);
```

```
do{
col--;
out(Rcol_pant,col);
out(Rdat_pant,'0'+(i&0x1));
out(Rcon_pant, 0x8000);
i=i>>1;
}while (col>0) }

.include “crt0.s”
.include “macro.s”
.data
w:                              .word 0x8888
.text
main:
$movei R1, w
LDB R1, 0(R1)
$CALL R6, mostra
HALT

mostra:
        $MOVEI R2, 16
        MOVI R3, 0
        MOVI R4, 0
        OUT Rfil_pant, R3
do:
        ADDI R2, R2, -1
        OUT Rcol_pant, R2
        $MOVEI R3, 0x1
        AND R3, R3, R1
        OUT Rdat_pant, R3
        $MOVEI R3, 0x8000
        OUT Rcon_pant, R3
        SHlL R1, R1, -1
        CMPGT R5, R2, R4
```

```
BNZ, do
JMP R6
```

En segon lloc, volem extendre el programa anterior, perquè es comporti així:

1) Mostrar el valor de w (el codi del programa anterior).

2) Esperar fins que es polsi una tecla.

3) Modificar w segons quina sigui la tecla polsada:

• Si la tecla és una 'A', fer un desplaçament lògic de w (SHL), 1 lloc a l'esquerra.

• Si és una 'B', fer un desplaçament lògic de w (SHL), 1 lloc a la dreta.

• Si és una 'C', fer un desplaçament aritmètic de w (SHA), 1 lloc a la dreta.

• Si és una 'D', s'ha de dividir per dos l'enter w (DIV).

• Si és un número n (tecles '0'. ... '9'), s'han de complementar els n bits de menys pes

de w. Nota: aquesta operació es pot escriure en C així: w = wˆ((1<<n)-1); on l'operador ˆ significa XOR bit a bit, i l'operador << significa desplaçament a Esquerra

4) Repetir des del pas número 1 (el programa no acaba, fa un bucle sense fi).

Completa l'exercici 4.8 abans de continuar:

Exercici 4.8: Escriu en alt nivell (en C) el nou programa principal (main), suposant que l'acció mostra és la que està declarada a la figura 4.1.

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

.data
w: .word 0xFFF8

.text
main:
    ; Cargar el valor de w desde la memoria
    $movei R1, w
    ld R1, 0(R1) ; R1 almacena el valor de w

    ; Sincronización con el teclado
```

```
    $movei R2, 0
    out Rcon_tec, R2

while_loop:
    ; Mostrar el valor de w
    $push R1
    $call R6, display_value
    $pop R1

    ; Esperar hasta que se pulse una tecla
do_wait:
    in R2, Rest_tec
    bz R2, do_wait

    ; Leer la tecla
    in R2, Rdat_tec
    $movei R3, tteclat
    add R2, R3, R2
    ldb R2, 0(R2) ; Valor de la tecla en ASCII

    ; Probar cuál es la tecla
    ; 1 - 'A
    $movei R3, 'A
    $cmpeq R3, R2, R3
    bz R3, else1
    $movei R4, 1
    shl R1, R1, R4
    bnz R3, end_if

else1:
    ; 2 - 'B
    $movei R3, 'B
    $cmpeq R3, R2, R3
    bz R3, else2
```

```
    $movei R4, -1
    shl R1, R1, R4
    bnz R3, end_if

else2:
    ; 3 - 'C
    $movei R3, 'C
    $cmpeq R3, R2, R3
    bz R3, else3
    $movei R4, -1
    sha R1, R1, R4
    bnz R3, end_if

else3:
    ; 4 - 'D
    $movei R3, 'D
    $cmpeq R3, R2, R3
    bz R3, else4
    $movei R4, 2
    div R1, R1, R4
    bnz R3, end_if

else4:
    ; 5 - Número (n = [1..9])
    ; if (0 <= R2 <= 9)
    $movei R3, '0
    $cmple R3, R3, R2
    bz R3, else5
    $movei R3, '9
    $cmple R3, R2, R3
    bz R3, else5

    ; Convertir el valor ASCII al valor numérico
    $movei R4, 48
```

```
    sub R2, R2, R4

    ; Complementar los bits de menor peso de w
    $movei R4, 1
    shl R2, R4, R2
    sub R2, R2, R4
    xor R1, R1, R2
    bnz R3, end_if

else5:
end_if:
    $movei R2, 0
    bz R2, while_loop

    HALT

display_value:
    $movei R2, 0
    out Rfil_pant, R2
    $movei R3, 16

display_loop:
    $movei R5, 0
    $cmpgt R5, R3, R5
    bz R5, end_display
    out Rcol_pant, R3

    ; Cargar el nuevo valor para mostrar en pantalla
    $movei R5, 0x0001
    and R5, R1, R5
    $movei R4, 48 ; Convertir a ASCII
    add R5, R5, R4
    out Rdat_pant, R5
    $movei R4, 0x8000
```

```
    out Rcon_pant, R4

    ; Shift lógico a la derecha
    $movei R5, -1
    shl R1, R1, R5
    addi R3, R3, -1
    bnz R5, display_loop

end_display:
    jmp R6
```