Nombre: Lucas Ricardo Williams
Padron: 103018

Ejercicio 1:

```
        .begin
        .org 2048
.macro push reg1
        add %r14, -4, %r14
        st reg1, %r14
.endmacro

.macro pop reg1
        ld %r14, reg1
        add %r14, 4, %r14
.endmacro

! hace el promedio entre 2 numeros, usa el %r5 y %r6 como auxiliares
! pre: la pila tiene 2 numeros
! post: deja en el tope de la pila el promedio entre los 2 numeros
avg:    pop %r5
        addcc %r5, %r6, %r6
        pop %r5
        addcc %r5, %r6, %r6
        srl %r6, 1, %r6
        push %r6
        jmpl %r15 + 4, %r0

main:   ld [x], %r1
        ld [y], %r2
        push %r1
        push %r2
        call avg
        pop %r3
        st %r3, [z]
        ld %r0, %r15
        jmpl %r15 + 4, %r0

x:      30
```

Acá asumís que %r6 está en cero en el inicio. Es válido para una simulación, pero en no en general

Los operandos los estás tomando de RAM, pero se pide que el programa reciba los operandos por pila.

```
y:      100
z:      0

        .end
```

Ejercicio 2:

```
IF R[IR[13]] THEN GOTO ----
IF R[IR[13]] THEN GOTO ----
R[temp0] <-- SEXT13(R[ir]);
R[temp0] <-- SIMM13(R[ir]);
R[pc] <-- ADD(R[rs1], R[rs2]); GOTO 0;
R[pc] <-- ADD(R[rs1], R[temp0]); GOTO 0;
R[rd] <-- ORCC(R[rs1], R[temp0]); GOTO 2047;
R[rd] <-- ORCC(R[rs1], R[rs2]); GOTO 2047;

op3 = 111000 (jmpl)
op3 = 010010 (orcc)

arithmetic formats
2b  5b   6b    5b           8b      5b
10| rd | op3 | rs1 | 0 | 00000000 | rs2
10| rd | op3 | rs1 | 1 | simm13 (13b)

microcodigo orcc
7: IF R[IR[13]] THEN GOTO 9;                    ! Branch si se debe realizar
operacion con numero instantaneo
8: R[rd] <-- ORCC(R[rs1], R[rs2]); GOTO 2047;   ! Realizo operacion orcc, guardo en
rd y salto a sig dir de programa
9: R[temp0] <-- SIMM13(R[ir]);                  ! La operacion simm llena de 0's
los 19 msb (most significant bytes)
10: R[rd] <-- ORCC(R[rs1], R[temp0]); GOTO 2047;! Realizo operacion orcc y salto a
sig dir de programa
```

OK

```
xxxxxx|x|xxxxxx|x|xxxxxx|x|0|0|xxxx|101|00001001
000000|1|000000|1|000000|1|0|0|0001|110|11111111
100101|0|000000|0|100001|0|0|0|1011|000|00000000
000000|1|100001|0|100101|0|0|0|0001|110|11111111

microcodigo jmpl
```

```
15: IF R[IR[13]] THEN GOTO 17           ! Branch si se debe realizar operacion con
numero instantaneo
16: R[pc] <-- ADD(R[rs1], R[rs2]); GOTO 0;      ! Realizo operacion ADD de rs1 y
rs2, guardo en pc y salto a linea 0 donde se realiza la operacion fetch
17: R[temp0] <--SEXT13(R[ir]);                  ! Extiendo los 19 bits de la cte
con la operacion SEXT13 (sign extension 13)
18: R[pc] <-- ADD(R[rs1], R[temp0]); GOTO 0;    ! Realizo operacion add entre rs1 y
temp0, guardo en pc y salto a linea 0 donde se realiza operacion fetch
```

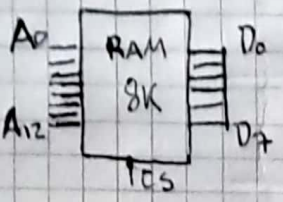
OK, muy bien

```
xxxxxx|x|xxxxxx|x|xxxxxx|x|0|0|xxxx|101|00010001
000000|1|000000|1|100000|0|0|0|0011|110|00000000
100101|0|000000|0|100001|0|0|0|1100|000|00000000
000000|1|100001|0|100000|0|0|0|0011|110|00000000
```

TP Ej 3 - Memoria

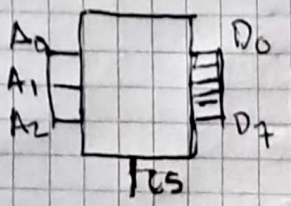
RAM disponible
8K × 8 bits

$2^{13} = 8K$

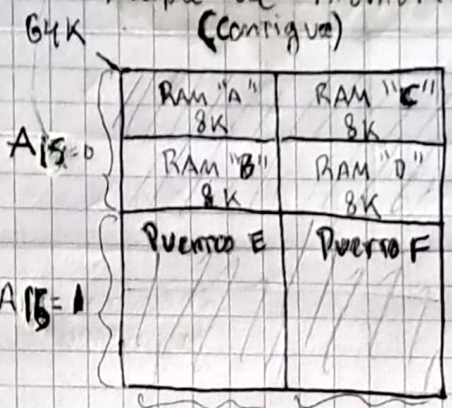


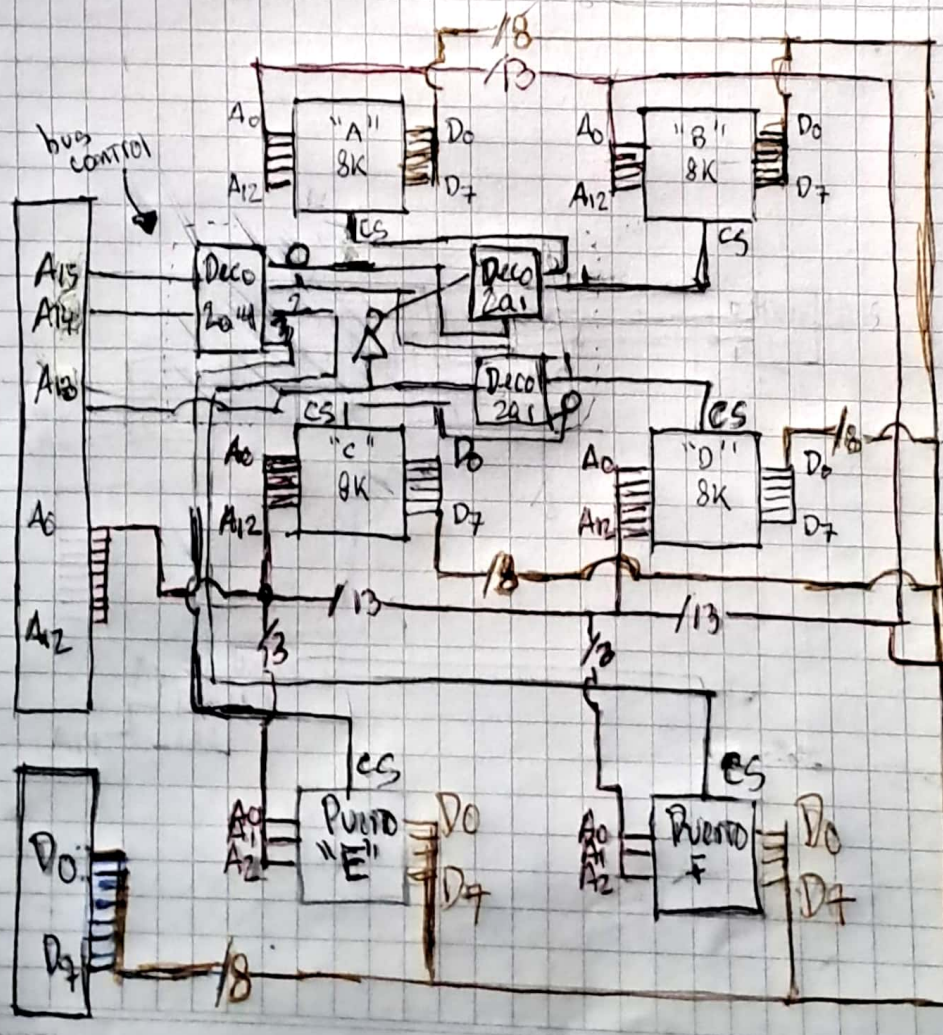
Procesador
16 bits
- direcciona 64K
- 8 bits ancho palabra

Puertos



Mapa de memoria (contigua)





Muy bien.Una forma más simple para resolver esto es con solamente un deco de 3 a 8.

NOTA

Rango de direcciones

bit	A_{15}	A_{14}	A_{13}	A_{12} ... A_0
RAM "A"	0	0	0	0 ..... 0 / 1 1
RAM "B"	0	0	1	0 ..... 0 / 1 1
RAM "C"	0	1	0	0 ..... 0 / 1 1
RAM "D"	0	1	1	0 ..... 0 / 1 1
Puerto "E"	1	0	X	X ... X $A_2A_1A_0$ 000 / 111
Puerto "F"	1	0	X	X ... X 000 / 111

Memoria fantasma