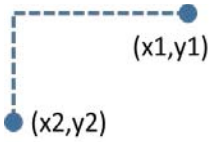
	Introduction to Computers **June 11, 2018. Partial exam (second term) / Final exam (part 2)**
	Name:_____ DNI: _____ Surname: _____

Students taking the Partial exam must answer all the questions.

Students taking the Final exam must answer questions: 1a, from 2a to 2e, 4a and 4b.

Exercise 1. When drawing the connections in an integrated circuit, the wires typically cannot follow diagonal paths; they must follow just horizontal and vertical paths on the circuit. In consequence, the length of the wire needed to connect two points of coordinates (x1,y1) and (x2,y2), respectively, is given by the expression Mdist = |x1-x2| + |y1-y2|, which is called, for obvious reasons, "the Manhattan distance" from (x1,y1) to (x2,y2), and can be denoted as Mdist(x1,y1,x2,y2). The length of the dotted line in the figure represents this value.



a) **(1,5 points)** Assume that we have two vectors, A and B, composed of 2N integers each. In both vectors, every pair of consecutive elements represents the coordinates of one point. We want to build a new vector C, of N integers, where every element of C represents the Manhattan distance between the two corresponding points of A and B, i.e. C[i] = Mdist(A[2*i], A[2*i +1], B[2*i] B[2*i +1]).
The vector C can be built with the following high-level pseudo-code:

```
#define N, ...
//Two operand vectors of N points
// A=[xa0,ya0,xa1,ya1,...] and B=[xb0,yb0,xb1,yb1,...]
int A[2*N] = {a list of 2N integers};
int B[2*N] = {a list of 2N integers};
//Result vector of N integers C=[c0,c1,...]
int C[N];   //Vector of distances
void main(void)
{
  int i;
  for (i=0; i < N; i++)
     C[i] = Mdist(A[2*i], A[2*i + 1], B[2*i], B[2*i + 1]) ;
}
```

Write an ARM assembly program that implements the described behavior, using the subroutine *Mdist*. This subroutine receives four signed integer parameters that represent the coordinates *(x1,y1)* and *(x2,y2)* of two points and returns the Manhattan distance between them.

b) **(1,5 points)** Follow the high-level pseudo-code given below to write the ARM assembly code of the subroutine *Mdist.* This subroutine must call another subroutine, *abs*, that computes the absolute value of an integer (you are not required to write the subroutine *abs*). Please, take into account that *Mdist* is not a leaf subroutine.

```
int Mdist(x1, y1, x2, y2)
{
  int d;
  d = abs(x1-x2)+ abs(y1-y2);
  return d;
}
```

REMARK: In all cases, the management of subroutines must be compliant with the standard studied in class.

Exercise 2. We want to run the following ARM assembly program. In the linker script we declare that the *.data* section is placed in the memory starting at address 0x0090, the *.bss* section is placed immediately after the *.data* section, and finally, the *.text* section goes immediately after the *.bss* section. The memory hierarchy

is composed of a Main Memory of 64 Kbytes and a direct cache ($) of 128 bytes. The block size is 16 bytes. Answer the following questions, explaining the reasoning that supports your solutions:

a) **(0.25 points)** Show the address format for the MM and the $.

b) **(0.5 points)** Find the range of memory addresses occupied by the array A. The same for the array B. How many memory blocks are occupied by the data? List the block addresses of such blocks.

c) **(0.25 points)** Find the range of memory addresses occupied by the program (plus the words that specify the initial addresses of the vectors A and B). How many memory blocks are occupied? List the block addresses of such blocks.

d) **(0.5 points)** For each $ block, show the value of the TAG field, and explain what information is stored in the block, when the first iteration of the loop (R2=0) is completed. Explain in which order the $ blocks are filled.

e) **(0.5 points)** Find the total number of $ misses that are produced when we execute the whole program.

```
.global start
.equ  N, 8
.data
A:      .word 12,3,13,14,5,9,0,1

.bss
B:      .space 4*N

.text
start:  ldr   R0, =A
        ldr   R1, =B
        mov   R3, #N
        sub   R5, R3, #1
        mov   R2, #0
for:    cmp   R2, R3
        bge   end_l
        ldr   R4, [R0, R2, LSL#2]
        str   R4, [R1, R5, LSL#2]
        add   R2,R2,#1
        sub   R5,R5,#1
        b     for
end_l:  b     .
.end
```

f) **(1 point)** If we increase the size of both arrays to 16 elements, show the new mapping of the program and data blocks into the $. What are the parts of the program and data that compete for the same $ block? Find the number of misses that are produced during the first four iterations (R2=0,1,2,3).

Exercise 3. An engineer uses a simulator to analyze the execution of a program on a given computer with a clock frequency of 1 GHz. He/she observes that: the 50% of the total execution time is due to memory accesses, each memory access takes 2 clock cycles, and the CPI is 8. Later, he/she introduces a cache memory with an access time of 1 clock cycle and simulates the same program again, observing that the 0.5% of the memory accesses provoke a miss. Answer the following questions, providing the adequate reasoning:

a) **(0.5 points)** The performance in MIPS of the original computer (without the cache)

b) **(0.5 points)** The performance in MIPS of the computer with the cache, assuming that the miss penalty is 100 cycles.

Exercise 4. We want to extend the instruction set of the BasicMIPS with a new instruction, BRZ, that implements a branch to the address specified in the first operand register if the contents of the second operand register is 0:

BRZ rs, rt // If RF(rt)=0 then (PC ← RF(rs)) ; else PC ← PC + 4

The instruction is encoded as shown below:

000111	rs	rt	Not used

Please, complete the following tasks:

a) **(1 point)** Extend the state diagram to include the new instruction.

b) **(1 point)** Modify the datapath to allow the execution of the new instruction.

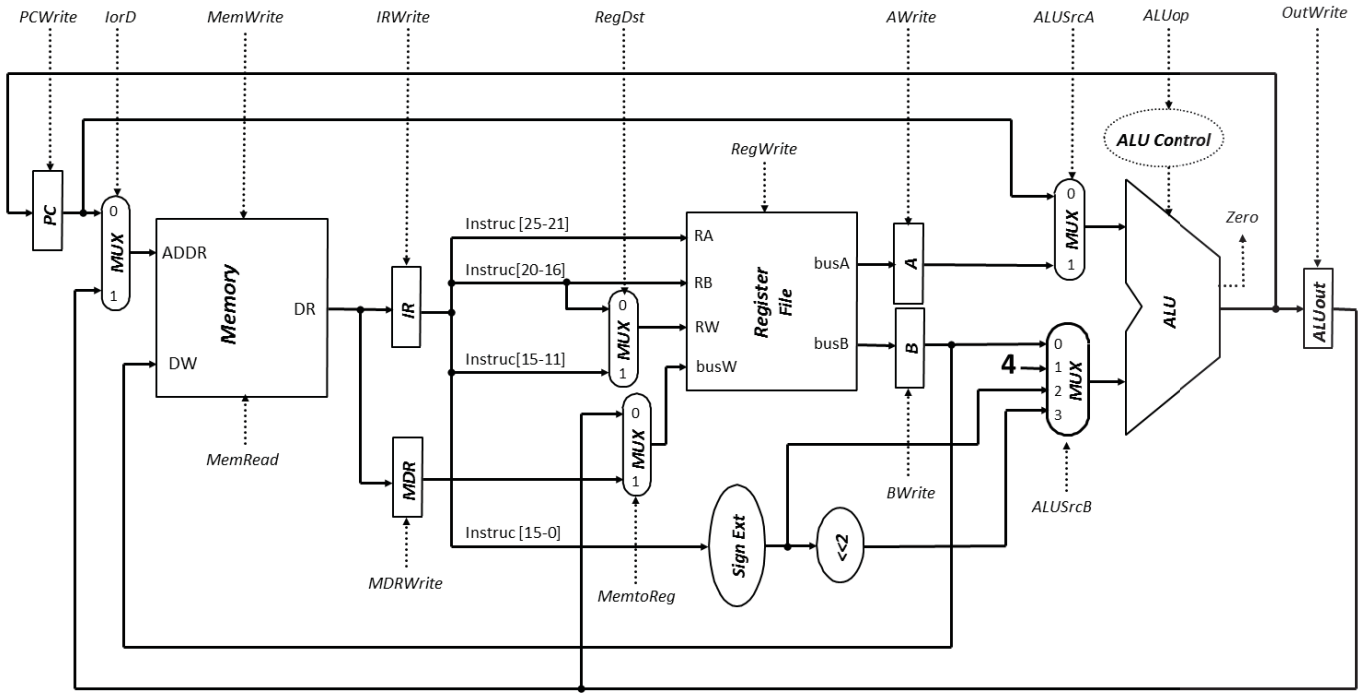
c) **(1 point)** Modify the truth table, adding any required rows and/or columns, in order to control the new datapath.

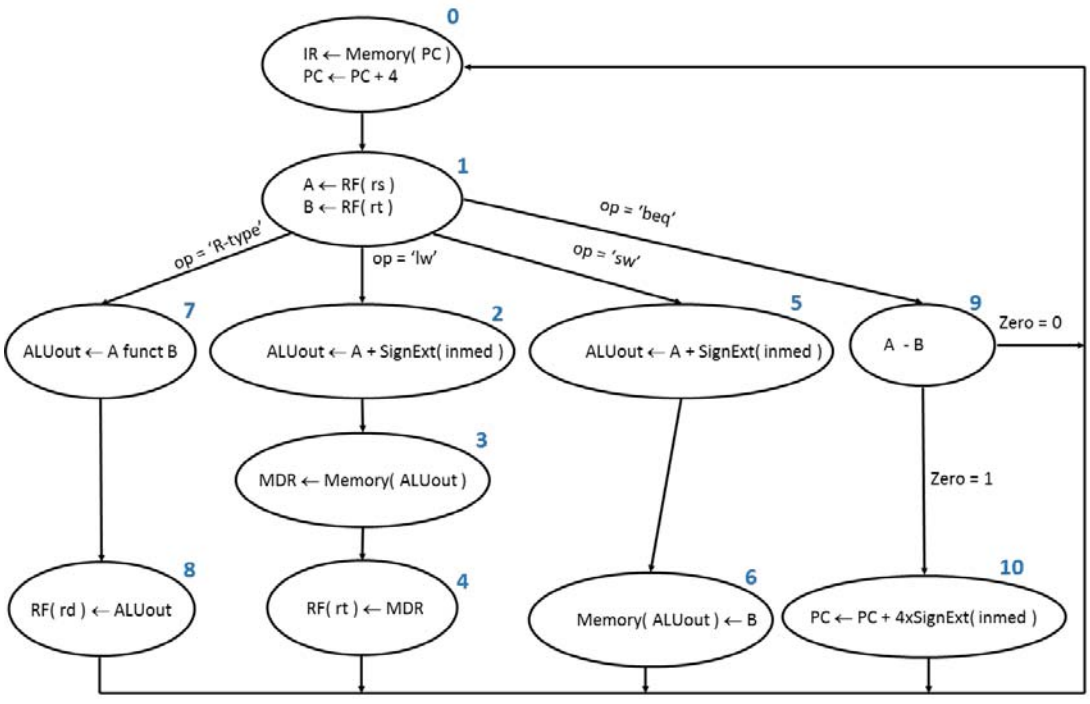
REMARK: For each part of this exercise, you must explain the reasons for your decisions.

Introduction to Computers
June 11, 2018. Partial exam (second term) / Final exam (part 2)

Name:_____ DNI: _____

Surname: _____





Current state		op	Zero	Next state	IRWrite	PCWrite	AWrite	BWrite	ALUSrcA	ALUScrB	ALUOp	OutWrite	MemWrite	MemRead	IorD	MDRWrite	MemtoReg	RegDest	RegWrite
0000	(fetch)	XXXXXX	X	0001	1	1			0	01	00 (add)		0	1	0				0
0001	(deco)	100011 (lw)	X	0010															
0001	(deco)	101011 (sw)	X	0101															
0001	(deco)	000000 (R-type)	X	0111	0	0	1	1					0	0					0
0001	(deco)	000100 (beq)	X	1001															
0010	(ex-lw)	XXXXXX	X	0011	0	0			1	10	00 (add)	1	0	0					0
0011	(mem-lw)	XXXXXX	X	0100	0	0							0	1	1	1			0
0100	(wb-lw)	XXXXXX	X	0000	0	0							0	0			1	0	1
0101	(ex-sw)	XXXXXX	X	0110	0	0		0	1	10	00 (add)	1	0	0					0
0110	(wb-sw)	XXXXXX	X	0000	0	0							1	0	1				0
0111	(ex-R)	XXXXXX	X	1000	0	0			1	00	10 (funct)	1	0	0					0
1000	(wb-R)	XXXXXX	X	0000	0	0							0	0			0	1	1
1001	(ex-beq)	XXXXXX	0	0000	0	0			1	00	01 (sub)		0	0					0
1001	(ex-beq)	XXXXXX	1	1010															
1010	(wb-beq)	XXXXXX	X	0000	0	1			0	11	00 (add)		0	0					0