	Introduction to Computers **June 19, 2017. Partial exam (second term)**
	Name:_________________________ DNI: ____________ Surname: ___________________________________

Exercise 1 (3 points). Given a vector, A, of N positive integers, we want to copy into a new vector, B, those elements of A that are prime numbers, according to the pseudo-code shown in the left box. The program calls the function `is_prime(int n)`, which returns the value 1 if the input argument is a prime number; otherwise it returns a 0. The pseudo-code for this function appears in the right box. The function `exact_div(n,i)` returns the value 1 if `n` is divisible by `i`, and 0 otherwise.

```
#define N 12
int A[N] = {A list of N values};
int B[N];
int i,j;

j = 0;
for (i = 0; i < N; i++) {
   if (is_prime(A[i]) == 1) {
      B[j] = A[i];
      j++;
   }
}
```

```
int is_prime(int n)
{
   int undecided = 1;
   int result = 0;
   int m;
   int i = 1;

   m = n/2;
   while (undecided == 1) {
      i++;
      if (i > m) {
         undecided = 0;
         result = 1;
      } else if (exact_div(n, i) == 1){
         undecided = 0;
      }
   }
   return result;
}
```

a) **(1.5 points)** Write an ARM assembly program that implements the program of the left box.

b) **(1.5 points)** Write an ARM assembly code for the subroutine `is_prime`, according to the specification given in the right box. We assume that the function `exact_div` has been previously implemented.

REMARK: All the code of this exercise must be compliant with the procedure call standard studied in class. The variables can be located in registers or memory at your own convenience.

Exercise 2 (1 point). An engineer has used a simulator to analyze the execution of a program on a given computer, assuming that each memory access takes just 1 cycle. After the simulation, he/she found that the CPI was 4.
a) (**0.5 points**) If the clock frequency of the computer were 1 GHz, what performance would be achieved in MIPS?
b) (**0.5 points**) The engineer is interested in the performance with a more realistic memory system, in which not all the memory accesses take just one cycle. After having simulated the same code with a new simulator that detects cache misses, he/she observes that 2% of instructions provoke a miss. Assuming a miss penalty of 100 cycles, find the new value of the performance in MIPS.

Exercise 3 (3 points). The left side of the box below shows an ARM assembly code, while the right side shows the corresponding disassembly in an ARM simulation environment. In the disassembly window, memory addresses are in hexadecimal, and we know that the content of the address 0c000048 is 0c000000, also in hexadecimal.

```
.global start
.EQU  N, 8
.data
A: .word 7,3,25,4,75,2,1,1
.text
start: mov r0, #0
       mov r1, #N-1
       ldr r2,=A
for1:  ldr r3,[r2,r0,lsl#2]
       sub r3,r3,#1
       str r3,[r2,r0,lsl#2]
       add r0,r0,#1
       CMP r0,#N
       BLT for1
       B .
.end
```

```
        A:
0c000000:   andeq r0, r0, r7
0c000004:   andeq r0, r0, r3
0c000008:   andeq r0, r0, r9, lsl r0
0c00000c:   andeq r0, r0, r4
0c000010:   andeq r0, r0, r11, asr #32
0c000014:   andeq r0, r0, r2
0c000018:   andeq r0, r0, r1
0c00001c:   andeq r0, r0, r1
        start:
0c000020:   mov r0, #0
0c000024:   mov r1, #7
0c000028:   ldr r2, [pc, #24]      ; [0xc000048]
        for1:
0c00002c:   ldr r3, [r2, r0, lsl #2]
0c000030:   sub r3, r3, #1
0c000034:   str r3, [r2, r0, lsl #2]
0c000038:   add r0, r0, #1
0c00003c:   cmp r0, #8
0c000040:   blt 0xc00002c <for1>
0c000044:   b 0xc000044 <for1+24>
0c000048:   stceq 0, cr0, [r0], {0}
```

The computer can address a Main Memory (MM) of 4 Gigabytes, and has a direct mapped cache ($) of 64 bytes, organized in blocks of 16 bytes. As can be seen in the disassembly window, data are loaded from address 0x0c000000, and the `.text` section begins immediately after the data. Please, answer the following questions:

a) **(0.5 points)** Show the address format for the MM and the $.

b) **(0.75 points)** List the MM blocks occupied by the data and the program, showing, for each of them, the corresponding $ block and tag value.

c) **(0.25 points)** Find the total number of memory accesses that will be generated by the execution of the code, from the *`start`* label until the first execution of the instruction *`b 0xc000044`*

d) **(1 point)** For the sequence of instructions of the previous question, find the number of $ misses. Show the contents of the $ directory (tag array) at the end of the execution.

e) **(0.5 points)** Assuming that the MM access time is 20 ns, the $ access time is 2ns, and the miss penalty is 50 ns, find the speed gain which is obtained in comparison to the same computer without the $.

Exercise 4 (3 points). We want to extend the instruction set of the BasicMIPS with the new instruction ADDRRI (Add Registers plus Immediate):

```
      ADDRRI Rt, Rs, Immediate
```

which is encoded as shown below:

6	5	5	16
110011	rs	rt	Immediate

and whose behavior is as follows:

```
      RF(rt) <- RF(rs) + RF(rt) + SignExt(Immediate), PC <- PC + 4
```

Please, complete the following tasks:

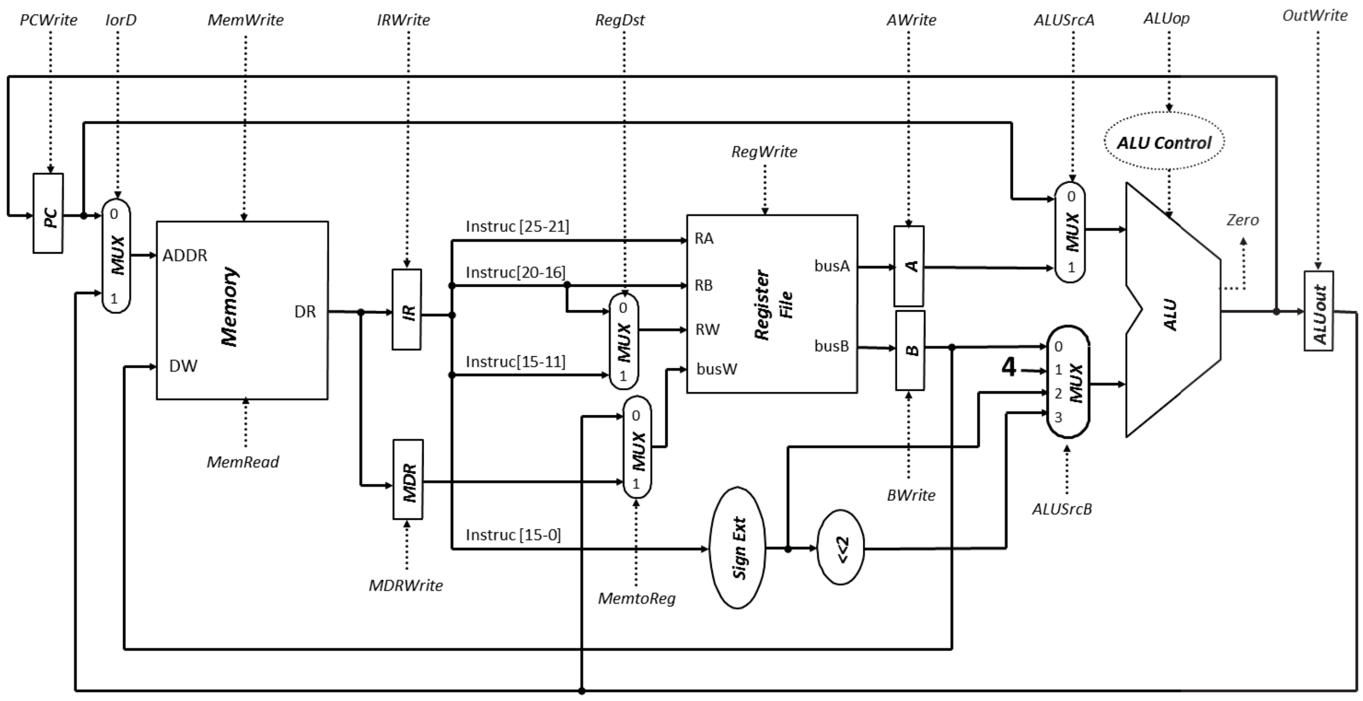
a) **(1 point)** Modify the datapath to allow the execution of the new instruction.

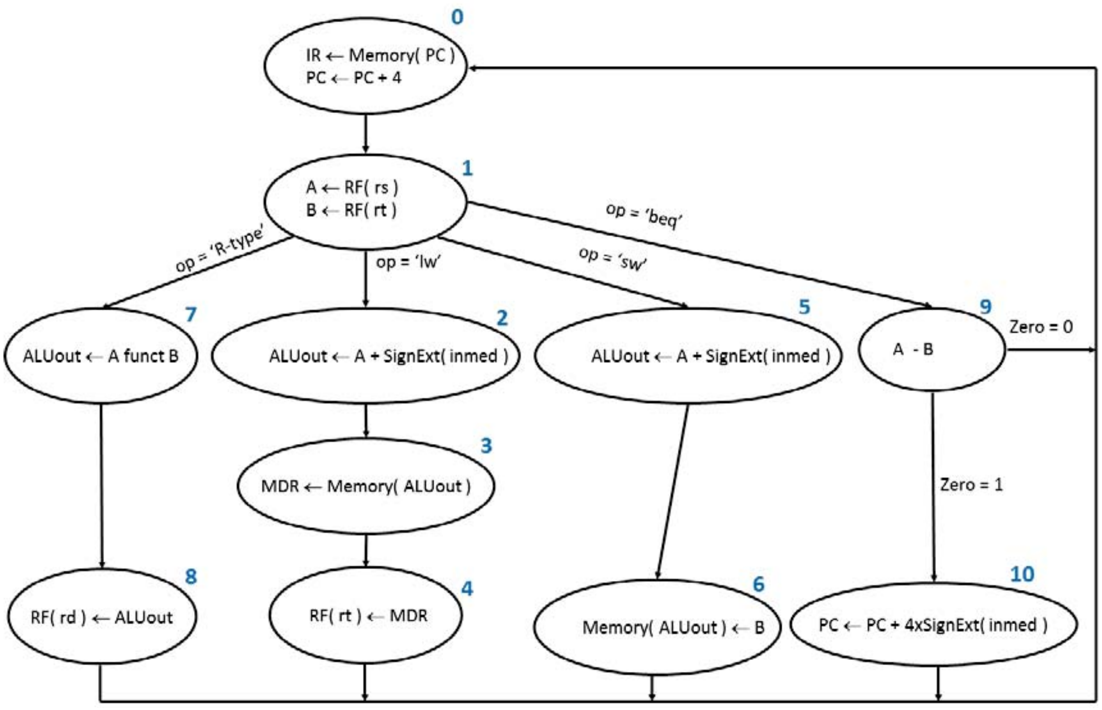
b**) (1 point)** Modify the state diagram to include the new instruction.

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 19, 2017. Partial exam (second term)**
	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	0	0	1	1					0	0					0
0001	(deco)	101011 (sw)	X	0101															
0001	(deco)	000000 (R-type)	X	0111															
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