Nom: _____

Puntuació: _____ / _____

Lab 2b - Castellano

Part 1

Alumno1
Nombre ____ Apellido1 ____ Apellido2 ____

Alumno2
Nombre ____ Apellido1 ____ Apellido2 ____

Introdución

Ejecutar el program *ejemplo.s* y rellenar el contenido de la siguiente tabla:

instrucciones	stalls	ciclos totales	CPI
12	9	25	2,08

```
        .ireg 15,60
        .data
a:      .dword  100
b:      .dword  0
        .text
inicio:         bnez r5,final
        dadd r3,r1,r2
        dsub r4,r1,r2
        and r5,r1,r2
        or r6,r1,r2
        xor r7,r1,r2
        ld r1,a(r0)
        sd r3, b(r0)
        sgt r2,r3,r4
        beqz r0,inicio ; salto incondicional
```

3 ciclos de parada en la bnez
3 ciclos de parada en la beqz
3 ciclos de parada en la bnez

10 instrucciones + 1(repeticion bnez) + trap

Ciclos totales = instrucciones + stalls + 4 (ciclos de carga hasta que se obtiene un resultado por ciclo

```
1       IF  ID  EX  M   WB
2           IF  ID  EX  M   WB
3               IF  ID  EX  M   WB
4                   IF  ID  EX  M   WB
5                       IF  ID  EX  M
WB
```

HASTA EL CICLO 5 NO TENGO NINGÚN

Detección y resolución de riesgos de datos mediar ciclos de parada

Introducir el código implementado en la `fase de decodificación`

```
// Riesgo entre EX e ID */
if (hay_fuente1_ID()
&& hay_destino_EX()
&&
IF_ID.IR.Rfuente1 ==
ID_EX.IR.Rdestino) {
IDstall = SI; IFstall = SI; }
else if (hay_fuente2_ID()
&& hay_destino_EX()
&& IF_ID.IR.Rfuente2 ==
ID_EX.IR.Rdestino) { IDstall = SI; IFstall = SI; }
```

```
// Riesgo entre MEM e ID
if (hay_destino_MEM()
&& hay_fuente1_ID()&&
EX_MEM.IR.Rdestino
==IF_ID.IR.Rfuente1) {
IDstall = SI; IFstall = SI;
}else if(hay_destino_MEM()
&&hay_fuente2_ID() &&
EX_MEM.IR.Rdestino==
IF_ID.IR.Rfuente2){
IDstall = SI; IFstall = SI; }
```

Completar la siguiente tabla con el contenido de los registros tras la ejecución de
datos1.s una vez incluido el código anterior:

R0	**R1**	**R2**	**R3**	**R4**
0	10	20	30	25

Detección y resolución de riesgos de datos en instrucciones aritméticas aplicando cortocircuitos

Introducir el código implementado en las funciones *mux_ALUsup*:

```
WBtoEX
if (hay_destino_WB()
 && hay_fuente1_EX()
 && MEM_WB.IR.Rdestino==ID_EX.IR.Rfuente1)
{ WBaEXalu_s = SI; result = wb; }
/ MEMtoEX
if (hay_destino_MEM()
&& hay_fuente1_EX()
&& EX_MEM.IR.Rdestino == ID_EX.IR.Rfuente1)
{ MEMaEXalu_s = SI; result = mem; } break;
```

Completar la siguiente tabla con el contenido de los registros tras la ejecución de *datos1.s* una vez incluido el código anterior:

R0	**R1**	**R2**	**R3**	**R4**
0	10	20	30	25

Detección y resolución de riesgos de datos en instrucciones de carga seguidas de aritmética aplicando cortocircuitos

Introducir el código implementado en las funciones:

`fase_decodificacion` y `mux_ALUsup`:

```
int fase_decodificacion (
case cortocircuito:
// Riesgo entre LD en fase EX
con otra en ID
if (hay_destino_EX()
&&hay_fuente1_ID()
&& IF_ID.IR.Rfuente1==
ID_EX.IR.Rdestino) {
IDstall = SI;
IFstall= SI;
}
```

Completar la siguiente tabla con el contenido de los registros tras la ejecución de `datos2.s` una vez incluido el código anterior:

R0	R1	R2	R3	R4
____	10	20	10	5

Resolución de riesgos de control mediante la estrategia *predict-not-taken*

Introducir el código implementado en la `fase de búsqueda`

```
void fase_busqueda (
case pnt:
 if (EX_MEM.cond) {
 Ifnop=SI;
 IDnop=SI;
EXnop=SI;
SaltoEfectivo = SI;
PCn = EX_MEM.ALUout;
} else {
PCn = PC+1; }
break;
```

Completar la siguiente tabla con el contenido de los registros y las posibiones de tras la ejecución de `suma.s` una vez incluido el código anterior:

R0	**R1**	**R2**	**R3**	**a**
0	6	40	0	6

Comparación entre estrategias de resolución de los riesgos de control:

	stalls	**predict-not-taken**
ciclos	____	____