Tarea entregable - Tomasulo

1. Mediante el simulador DLXview, realizar el siguiente ejercicio:

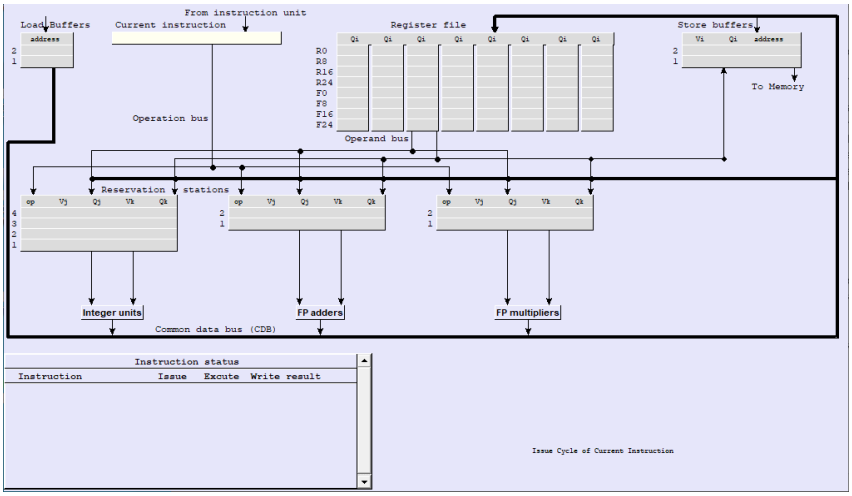
Sea un procesador que implementa el algoritmo de Tomasulo sin especulación con las siguientes características:

- Las instrucciones enteras tienen un tratamiento análogo a las instrucciones de punto flotante, para lo que:
    - Las unidades enteras tienen asociadas estaciones de reserva, donde las instrucciones aguardan a que se resuelvan sus dependencias LDE.
    - Las instrucciones enteras hacen uso del CDB para escribir sus resultados en el banco de registros enteros.
    - El TAG debe extenderse para considerar las ER enteras (int1, int2, etc.)

- Las latencias de las unidades son las siguientes (esto se debe configurar en DLXview utilizando el botón “configure” de la aplicación que aparece una vez ésta se abre, y tras haber seleccionado “Tomasulo” a continuación):
    - INT: latencia =1 ciclo
    - FPadd: latencia =4 ciclos
    - FPmul: latencia =6 ciclos
    - Load: latencia =1 ciclo
    - Store: latencia =1 ciclo

- Por ahora, el/la alumno/a se puede seleccionar el número de estaciones de reserva de cada tipo que desee. DLX asume que existe una unidad funcional asociada a cada estación de reserva de cada tipo, y que las unidades funcionales de suma y multiplicación están segmentadas. Se asume también que el delay slot de los saltos es de una instrucción.

- En el caso de las dependencias LDE, las instrucciones consumidoras no pueden ejecutarse en el mismo ciclo en el que la instrucción productora escribe en el CDB el dato que estaban aguardando.



Determinar el mínimo número de recursos (número de estaciones de reserva, de buffers de loads y de buffers de stores) que es necesario para que la etapa de issue del siguiente código no sufra ninguna parada al ejecutarse el siguiente código:

```
      ld    f2, a
      add   r1, r0, xtop
loop: ld    f0, 0(r1)
      sub   r1, r1, #8
      multd f4, f0, f2
      bnez  r1, loop
      sd    8(r1), f4
      trap  #0
```

Para ello se proporcionan los ficheros `ej1.d` y `ej1.s`, que es preciso seleccionar conjuntamente y cargar al mismo tiempo en el simulador (utilizando el botón "load", que es seleccionable una vez se haya configurado el procesador).

2. Mediante el simulador DLXview realiza el siguiente ejercicio:

Suponiendo el mismo procesador que en el ejercicio anterior (mismas latencias mostradas en el enunciado y el número mínimo de recursos obtenido), implementa un código DLX que realice, en el menor número de ciclos posible, la funcionalidad mostrada a continuación, que representa a una red neuronal de 5 entradas formada por 2 neuronas de tipo perceptrón:

```
float W[2][5]={0.3,0.45,1.2,6.8,3.2,1.1,0.8,2.2,1.5,0.68}; // matriz
de pesos
float B[2]={0.3,1.1}; //vector de sesgos
float U[2]={2.3,3.8}; // vector de umbrales
float X[5]={ 0.1,0.1,0.2,0.3,0.4}; //vector de entradas
float Y[2]; // vector de salidas

int main() {
int i,j;
float tmp;

for (i=0;i<2;i++) {
      tmp=B[i];
      for (j=0;j<5;j++) {
            tmp=tmp+W[i][j]*X[j];
      }
      if (tmp> U[i]) Y[i] = tmp;
      else Y[i] = 0;
}
return 0;
}
```

Nota: basta con proporcionar los ficheros `ej2.s` y `ej2.d` empleados.