Electrónica Digital 2
Programa de Ingeniería Electrónica
Universidad de Antioquia

Examen: tema 3 – microarquitectura de la CPU (10%)
Fecha: viernes 17 de mayo de 2024
Duración: una hora y cuarenta minutos (1h:40m)

SOLUCIÓN

EJERCICIOS PARA RESOLVER

1) (16pts) Afirmaciones para completar relacionadas con las microarquitecturas ARM de un solo ciclo y *pipelined* estudiadas en clase:
   a. (3p) Las instrucciones LSL y MOV pertenecen al grupo de instrucciones de **procesamiento de datos**.
   b. (3p) La microarquitectura ARM *pipelined* tiene **1** sumador(es) para generar el valor del registro R15.
   c. (3p) Un total de **4** flip-flop(s) se requieren en la unidad de control para almacenar las *flags* de la ALU.
   d. (3p) La etapa del *pipeline* donde se determina la dirección a la que se debe saltar cuando se procesa una instrucción de salto es **EJECUCIÓN**.
   e. (4p) Si en un programa escrito en ensamblador, el 30% de los LDRs requieren bloqueo (*stall*) en el procesador ARM *pipelined*, el CPI para dicha instrucción será de **0.7*1+0.3*2 = 1.3**.

2) (42pts) Problemas de análisis relacionados con las microarquitecturas ARM de un solo ciclo y *pipelined* estudiadas en clase:

   a. (20pts) Indique al frente de las siguientes instrucciones si se requiere de una o más instrucciones NOPs antes de la correspondiente instrucción, para garantizar una correcta y eficiente ejecución en un procesador ARM *pipelined* que no dispone de unidad de resolución de *hazards*:

	¿Require NOPs?	¿Cuántos?
`MOV R0, #1`		
`MOV R1, #2`		
`ADD R0, R1, R2`	**SI**	**2**
`LDR R3, [R1]`	**NO**	**0**
`CMP R0, R3`	**SI**	**2**
`BLS FINISH`	**NO**	**0**

   b. (8pts) ¿Qué instrucción o conjunto de instrucciones de las disponibles en el procesador ARM de un solo ciclo implementado en la práctica 5 se podrían emplear para reemplazar la funcionalidad de la instrucción BL?
   1) **`MOV LR, PC`** 2) **`B LABEL`**

   c. (14pts) Se implementó un procesador ARM *pipelined* con unidad de hazards que soluciona los problemas de dependencia de datos mediante *bubbles* en lugar de *forwarding*. Indique, para el siguiente código, el número de ciclos que le tomará al procesador ejecutar las instrucciones dadas, asumiendo que la primera instrucción se ejecuta después de un *reset*:

```
ADD    R0, R1, R2
SUB    R3, R1, R2
LDR    R1, [R0]
STR    R1, [R3]
```

   Número de ciclos con *bubbles*: **11 ciclos**

   Si se implementa la técnica de *forwarding*, indique la cantidad de ciclos para las mismas instrucciones:
   Número de ciclos con *forwarding*: **9 ciclos**

3) (42pts) Para el procesador ARM de un solo ciclo estudiado en clase e implementado en la práctica de laboratorio no. 5 (ver figura), complete la siguiente tabla (en binario o hexadecimal) de acuerdo con las instrucciones en ensamblador dadas. Tenga en cuenta que:
   - XYZ corresponde con los tres últimos dígitos de su documento de identificación (Ej. si Doc = 123456789, X=7, Y=8, Z=9. MOV R0, #0x78, MOV R1, #0xF9).
   - Cuando el valor de la correspondiente señal no importe, se deberá colocar una **x** (don't care) (Ej. RegSrc = 1x).
   - El procesador soporta las instrucciones: ADD, SUB, AND, ORR, LDR, STR, Bxx, además de CMP, LSL, LSR, ASR, ROR y MOV (con ALUControl = 3'b100 para los desplazamientos y el MOV).

Instrucción	RegSrc 2'b	RA1 4'b	OutSh 32'h	ALUSrc 1'b	ALUControl 3'b	ExtImm 32'h	ALUResult 32'h
MOV R0, #0xAC	0X	XXXX	XXXXXXXX	1	100	000000AC	000000AC
MOV R1, #0xF5	0X	XXXX	XXXXXXXX	1	100	000000F5	000000F5
LSLS R2, R1, #24	0X	XXXX	F5000000	0	100	XXXXXXXX	F5000000
ADDMI R2, R0, R2, ASR #24	00	0000	FFFFFFF5	0	000	XXXXXXXX	000000A1
ADDPL R2, R0, R1	00	0000	000000F5	0	000	XXXXXXXX	000001A1
END: B END	01	1111	XXXXXXXX	1	000	FFFFFFF8	00000014