Practica 4

Objectius

L'objectiu de la pràctica és visualitzar els diferents passos que ha de fer un microprocessador per tal d'executar una instrucció.

Explicació de la practica

Aquesta practica consisteix en fer un exercici guiat, i després l'informe on hi han 9 preguntes que per resoldre-les s'han de mirar els codis 3 i 5.

Exercici Guiat:

1) Abans d'executar els codis tracta d'esbrinar la seva funcionalitat. Faran el mateix? Creus què trigaran el mateix nombre de cicles en executar-se?

Els dos codis fan el mateix, guardar un quatre a memòria però amb unes quantes instruccions de més que, com a mínim, al primer no afecten. Al segon, en canvi, tenim un salt que depèn del valor a a7, salt que s'efectua igualment així que la funcionalitat acaba sent la mateixa. A causa d'aquest salt al segon codi, el temps d'execució d'aquest s'incrementa respecte del primer ja que, com que estem executant en pipeline de 5 cicles, per culpa del salt tenim un flush que ens fa perdre 2 cicles.

3) Executa els dos codis cicle a cicle fixant-te com les instruccions van passant per les diferents etapes del pipeline. Compta els nombre de cicles que es necessiten per executar cadascun dels codis i compara les Pipeline tables.

En el codi 1 es realitzen 12 cicles:

	0	1	2	3	4	5	6	7	8	9	10	11	12
add x13 x0 x0	IF	ID	EX	MEM	WB								
add x17 x0 x0		IF	ID	EX	MEM	WB							
addi x12 x0 4			IF	ID	EX	MEM	WB						
add x13 x12 x13				IF	ID	EX	MEM	WB					
addi x17 x17 -1					IF	ID	EX	MEM	WB				
auipc x10 0x10000						IF	ID	EX	MEM	WB			
addi x10 x10 -20							IF	ID	EX	MEM	WB		
sw x13 0(x10)								IF	ID	EX	MEM	WB	

En el codi 2 es realitzen 15 cicles:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
add x13 x0 x0	IF	ID	EX	MEM	WB											
add x17 x0 x0		IF	ID	EX	MEM	WB										
addi x12 x0 4			IF	ID	EX	MEM	WB									
add x13 x12 x13				IF	ID	EX	MEM	WB								
addi x17 x17 -1					IF	ID	EX	MEM	WB							
blt x17 x0 4 <salta>						IF	ID	EX	MEM	WB						
auipc x10 0x10000							IF	ID	IF	ID	EX	MEM	WB			
addi x10 x10 -24								IF		IF	ID	EX	MEM	WB		
sw x13 0(x10)											IF	ID	EX	MEM	WB	

Fins la instrucció blt tots dos codis fan el mateix, la cosa canvia a partir d'aquest punt. Mentre que el 1r codi avança sense problema, fent una instrucció per cicle, el 2n codi a la següent instrucció al blt podem veure com crida 2 vegades al Instruction Fetch a causa del salt, en tornar a cridar al IF els dos cicles d'abans "no serveixen" i s'han de tornar a fer, cosa que retarda tot el procés 2 cicles, més el cicle extra pel fet que hi ha 1 instrucció més, ens dona els 3 cicles que hi ha de diferència entre el 2n codi i el 1r.

4) Perquè les etapes de la instrucció auipc x10 0x 10000 al pipeline son IF ID IF ID EX MEM WB al Codi 2?

Això és degut a que el processador per intentar guanyar temps, encara que no sàpiga si haurà de saltar o no, el que fa és anar executant les següents instruccions després del salt predient el cas en el que no es fa el salt.

Per això mateix en el nostre cas com la condició si es compleix i ha de saltar, les 2 microinstruccions que havia fet per estalviar temps les ha d'esborrar i tornar a començar, per això veiem que torna a fer IF i ID. Les altres microinstruccions ja segueixen sense problemes.

5) Com afectaria al nombre total de cicles d'execució el següent canvi en el codi:

El nombre total de cicles s'incrementa d'11 a 14. Aquests 3 cicles de més són el cicle de la instrucció de salt en si, i els dos cicles que es perden a causa del flush.

Informe:

1) Quin es l'estat de cadascuna de les cinc etapes del pipeline al cicle 6? I al 8?

- Codi 3:

Cicle 6:

Instruction Fetch (IF): la a0, resultat

Instruction Decode (ID): bgt a7, zero, salta

Execute (EX): addi a7, a7, -1

Memory Access (MEM): add a3, a2, a3

Writeback (WB): addi a2, zero, 4

Cicle 8:

Instruction Fetch (IF): sw a3, 0(a0)

Instruction Decode (ID): la a0, resultat, en concret la segona part que es: addi x10 x10 -24

Execute (EX): la a0, resultat la primera part que es: auipc x10 0x10000

Memory Access (MEM): bgt a7, zero, salta

Writeback (WB): addi a7, a7, -1

-Codi 5:

Cicle 6:

Instruction Fetch (IF): addi a7, a7, -1

Instruction Decode (ID): add a3, a2, a3

Execute (EX): add a3, zero, zero

Memory Access (MEM): addi a2, zero, 9

Writeback (WB): lw a7, 0(a0)

Cicle 8

Instruction Fetch (IF): la a0, guardaResultat

Instruction Decode (ID): bgt a7, zero, loop

Execute (EX): addi a7, a7, -1

Memory Access (MEM): add a3, a2, a3

Writeback (WB): add a3, zero, zero

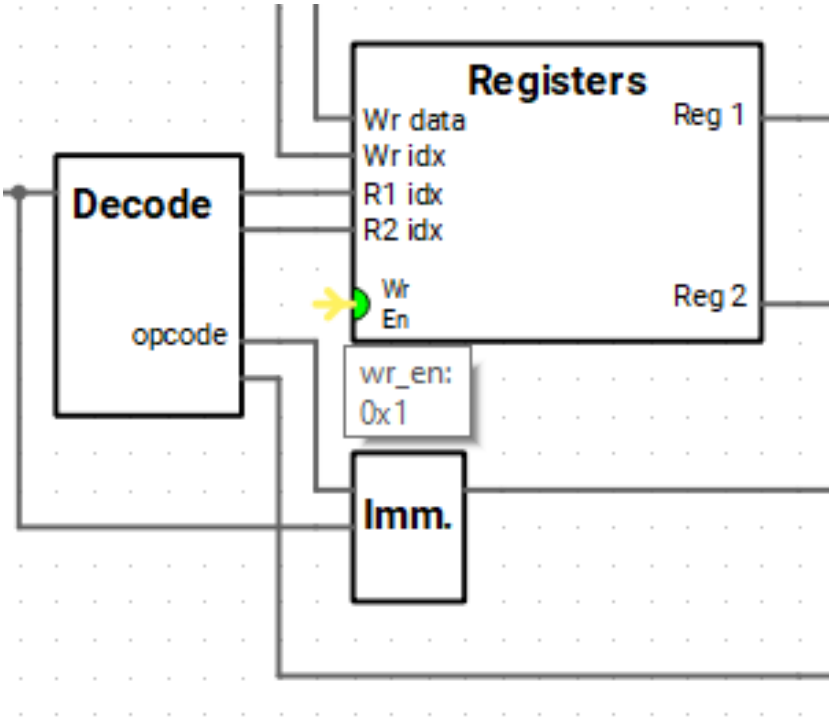
2) Quins senyals de control s'activen en el cicle 4 a la segona etapa del pipeline? A quines instruccions del codi corresponen?

Codi 3:

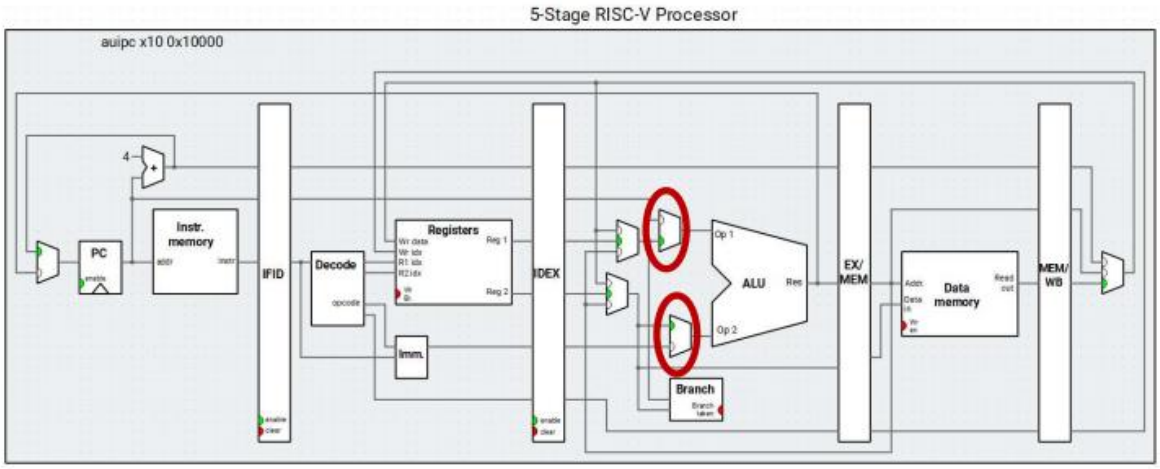
Al cicle 4 a la segona etapa del pipeline s'activa la senyal de control del banc de registres, en aquest cas la senyal permet l'escriptura. Correspondrà a la instrucció add a3, a2, a3.

Codi 5:

Al cicle 4 a la segona etapa del pipeline s'activa la senyal de control del banc de registres, en aquest cas la senyal permet l'escriptura. Correspondrà a la instrucció add a2, zero, 9.

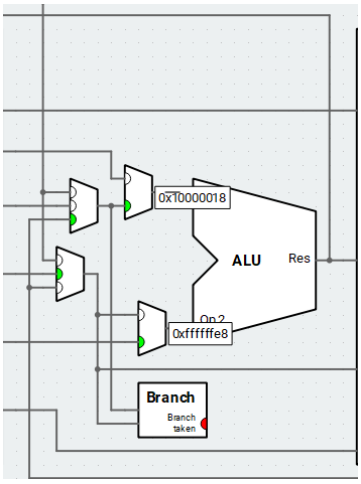


3) Quins són els valors a les sortides dels multiplexors assenyalats a la figura al cicle 9:



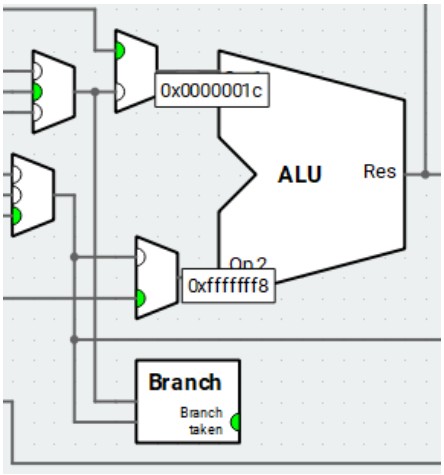
Codi 3:

Els valors a les sortides dels multiplexors son: $0x10000018_h$ i $0xfffffffe8_h$



Codi 5:

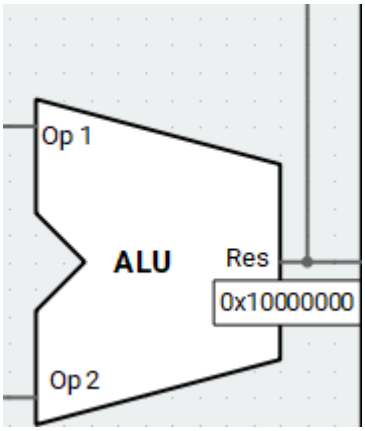
Els valors a les sortides dels multiplexors son: $0x1000001c_h$ i $0xffffffff8_h$



4) Què està calculant la ALU al cicle 9?

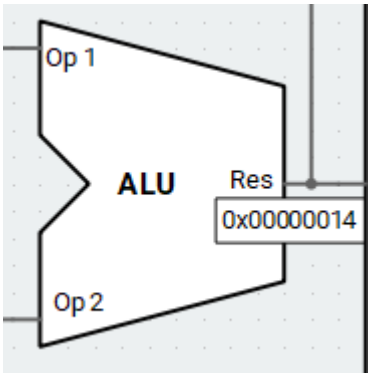
Codi 3:

La ALU esta calculant els 12 bits menys significatius de la direcció de memòria on esta l'etiqueta resultat per sumar-los als 20 més significatius i així tenir l'adreça completa de 32 bits.

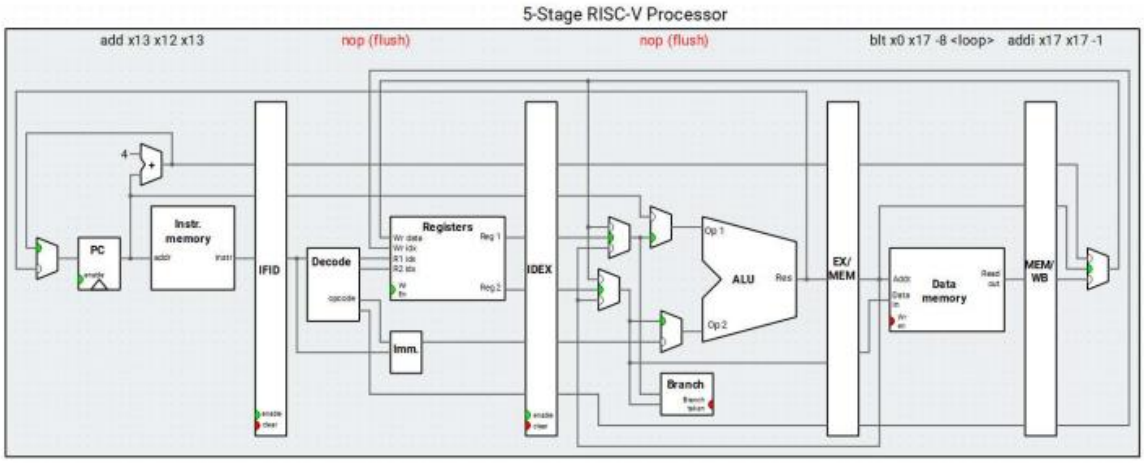


Codi 5:

La ALU esta calculant els 12 bits menys significatius de la direcció de memòria on esta l'etiqueta loop per sumar-los als 20 més significatius i així tenir l'adreça completa de 32 bits.



5) Llegeix amb cura la part del codi amb que s'implementa el loop. Tenint en compte el que has vist a la qüestió 3), justifica perquè el pipeline al cicle 10 presenta aquest estat:



Codi 5:

En el cicle 10, el pipeline hi ha un flush ja que quan entra la instrucció del salt condicional el programa va avançant per no quedar-se esperant per veure si el salt es compleix o no , per això quan arriba a la tercera etapa, el execute (EX), i es comprova que el salt si que es fa, s'han de borrar les instruccions que s'havien executat prèviament. Llavors el fetch apunta a la primera instrucció que s'executarà després de l'etiqueta loop, en aquest cas add a3, a2, a3

6) Quan NO es produeix el salt, quantes etapes de la instrucció auipc x10 0x 10000 s'executen al pipeline?

Quan no es produeix el salt s'executen les 5 etapes al pipeline ja que tot el flux d'execució passa per aquesta instrucció.

7) Quan s'està executant la instrucció de salt (etapa EX), però el salt NO es produeix, a quina posició apunta el program counter (PC)?

El PC apunta a la posició $0x00000024_h$.

8) Quan es produeix el salt, quantes etapes de la instrucció auipc x10 0x 10000 s'executen al pipeline?

Quan es produeix el salt s'executen dues etapes al pipeline, la Instruction Fetch (IF) i la Instruction Decode (ID) ja que quan la instrucció del salt condicional arriba al execute.

9) Quan s'està executant la instrucción de salt (etapa EX), i el salt es produeix, a quina posició apunta el program counter (PC)?

El PC apunta a la posició $0x00000024_h$.

Conclusió

En aquesta practica 4, he après sobre el pipeline i les microinstruccions realitzades al 5 Stage RISC-V Processor que era una cosa que hem costava molt d'entendre, i crec que ara ja m'ha quedat tot més clar.

També he vist com s'executa el flush i que realitza cada etapa en aquest moment, tant com si es compleix la condició del salto com si no es compleix.