UNIVERSITAT DE BARCELONA

Pràctica 5: Introducció al simulador i8085

Alejandro Guzman

29 Abril 2022

ÍNDEX

1 OBJECTIUS

Aquesta és la primera pràctica que realitzarem amb el simulador i8085.

L'objectiu principal de la pràctica és, de la mateixa manera que la primera pràctica del simulador Ripes, serà aprendre a utilitzar i analitzar tot l'entorn relacionat amb aquest nou simulador.

Aprendrem a analitzar el *pipeline* per a poder veure l'estat de l'execució de les diverses instruccions del programa a cada cicle, així com diversos fenòmens que es poden donar degut a dependències (retard d'execució o *stall*) i altres canvis al codi que afecten directament sobre el temps d'execució degut a la variació del nombre de cicles.

També aquesta darrera pràctica serà vital per veure la importància que té en el temps d'execució el fet de no utilitzar pipeline o sí utilitzar-lo realitzant una comparativa amb un programa en *Risc-V* en *Single Cycle* o *Multi-Cycle*.

2 EXERCICIS DEL GUIÓ DE LA PRÀCTICA

1 Qüestió 1

Quin es l'estat de cadascuna de les cinc etapes del *pipeline* al cicle 6?
L'etapa IF (*instruction fetch*) es troba amb la instrucció de suma aritmètica d'inmediat *addi x17 x17 -1*, l'etapa ID (*instruction decode*) es troba amb la instrucció de suma aritmètica de registres *add x13 x12 x13*, l'etapa EX (*execute*) es troba amb la instrucció de suma aritmètica de registres *add x13 x0 x0*, l'etapa MEM (*memory write*) es troba amb la instrucció de suma aritmètica d'inmediat *addi x12 x0 9* i per últim l'etapa WB (*write back*) es troba amb la instrucció de *load word* (càrrega de dataa registre) *lw x17 0(x10)*.

Quin es l'estat de cadascuna de les cinc etapes del *pipeline* al cicle 8?
De la mateixa manera, al cicle 8 la instrucció *auipc x10 0x10000* es carrega, *blt x0 x17 -8 <loop>* es descodifica, *addi x17 x17 -1* s'executa, add x13 x12 x13 realitza els accessos a memòria (tot i que no és necessari) i, finalment, *add x13 x0 x0* escriu el resultat de l'operació realitzada al registre x13.

2 Qüestió 2

Quins senyals de control s'activen en el cicle 4 a la segona etapa del pipeline? A quines instruccions del codi corresponen?
En el cicle 4 el processador es troba de la següent forma:
En el cicle 4 i en la segona etapa s'activa el bit de *write-enable* que permet escriure al banc de

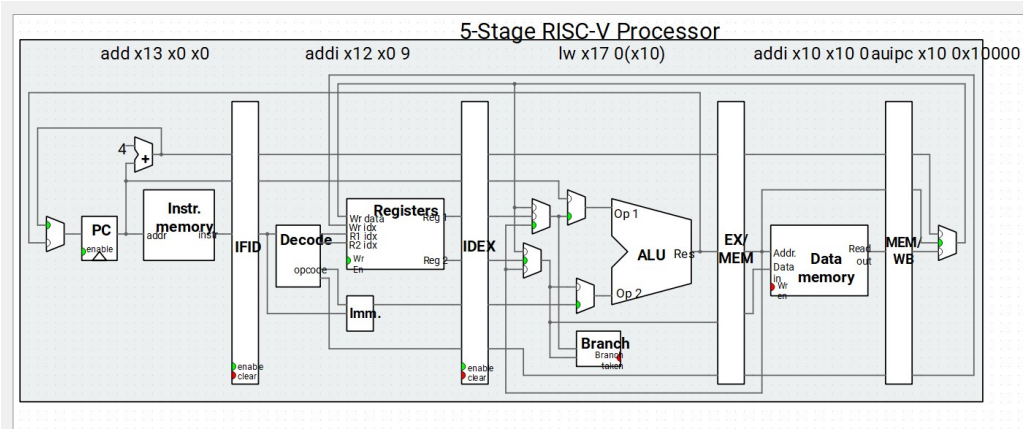

Figura 2.1: Estat del processador en el cicle 4

registres, que, tot i ser part d'aquesta segona etapa, correspon a l'última de la instrucció *auipc x10 0x10000* (primera part de la pseudoinstrucció *la a0, valorDada*), que es troba a la fase de *Writeback*. És a dir, al registre especificat (x10) es guarda el resultat de l'operació realitzada per la ALU ([PC] + (0x10000 « 20)) síncronament. També s'activa el senyal d'enable de IDEX per tal de guardar els valors generats en aquesta etapa de la instrucció *addi x12 x0 9* (*addi a2, zero, 9*) síncronament i usar-los a la següent etapa (la d'execució).

3 Qüestió 3

Quins són els valors a les sortides dels multiplexors assenyalats a la figura al cicle 9
Com es pot comprovar en el cicle 9 s'està executant la instrucció *blt x0 x17 -8 <loop>*. Llavors en el primer multiplexor s'esta seleccionant el valor del PC que, com és la octava instrucció,

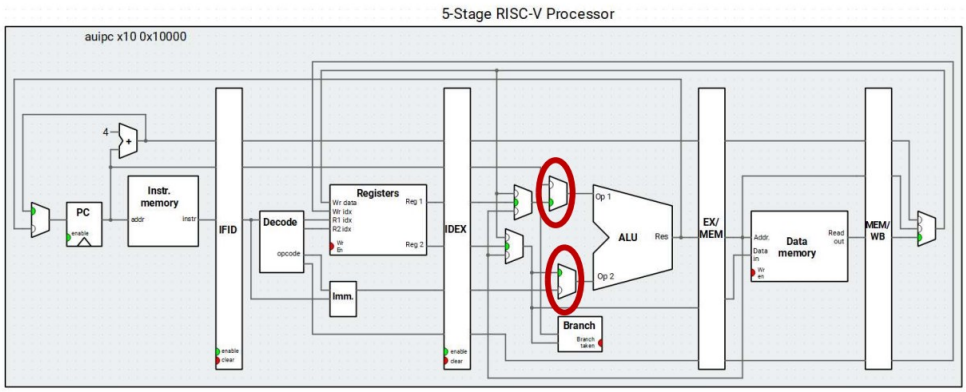

Figura 2.2: Processador amb dos multiplexors assenyalats

el seu valor és $4 \cdot 7 = 28 = 0x1c$. El segon multiplexor val 0xfffffff8 = (-8), ja que per calcular l'adreça de salt <*loop*> el processador realitza un adreçament relatiu sumant al contingut del PC l'immediat corresponent (obtingut a través de la descodificació de la instrucció).

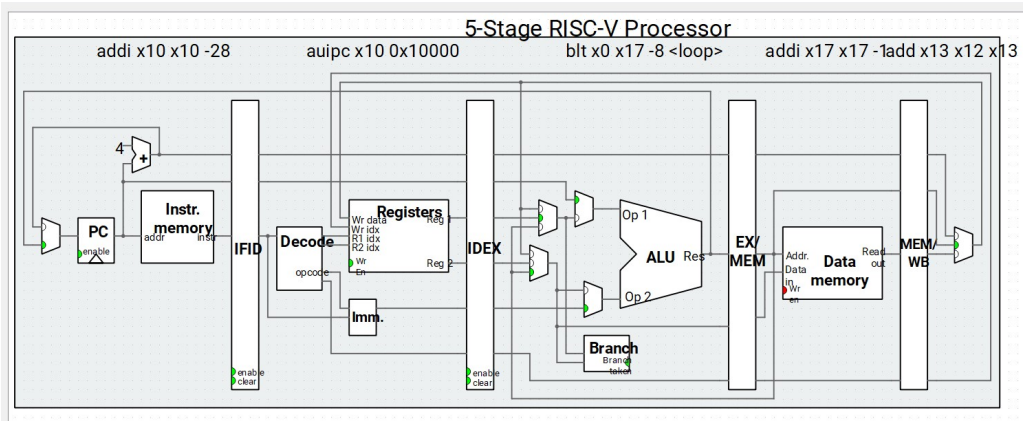

Figura 2.3: Estat del processador en el cicle 9

4 Qüestió 4

Què està calculant la ALU al cicle 9?
Tal com hem mencionat, es calcula l'adreça de salt de la instrucció (es tracta d'un salt condicional) mitjançant un adreçament de memòria relatiu, és a dir, suma al contingut del PC, al carregar la instrucció, un immediat determinat a l'ensamblatge del programa per tal de determinar-la correctament. Si es compleix la condició de salt (que és el cas), aquest valor es carregarà al PC.

5 Qüestió 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:
Justament acabem de realitzar un salt al cicle anterior. Per tant, les dues instruccions que s'estaven carregant i descodificant (*addi x10 x10 -28* a Fetch i *auipc x10 0x10000* a Decode) no s'han d'executar i no es pot continuar amb la següent etapa d'ambdues instruccions. A

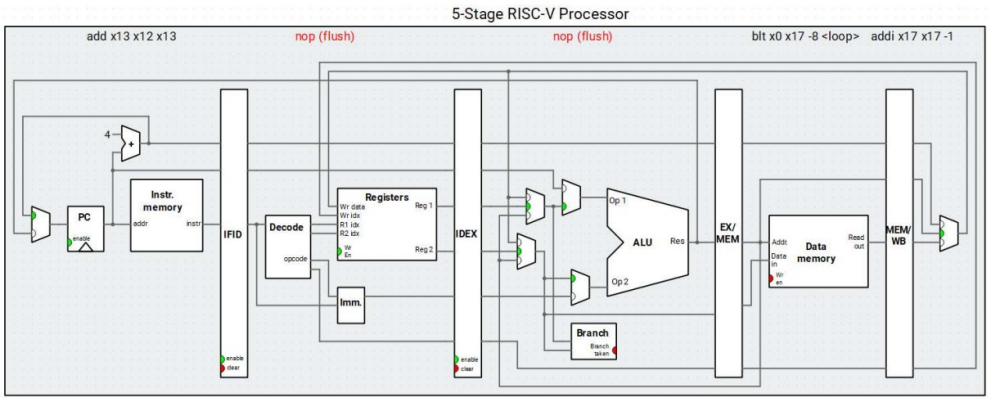

Figura 2.4: Estat del pipeline al cicle 10

causa d'això s'introdueix al *pipeline* dues etapes d'espera *nop*. El processador seguirà el seu funcionament normal a partir del salt realitzat, sempre i quan no es realitzi un altre salt.

6 Qüestió 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 el processador funciona amb normalitat, el que significa que cada instrucció, incluïda la esmentada, s'executa en les cinc etapes IF, ID, EX, MEM i WB.

7 Qüestió 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)?
Si no es produeix el salt, llavors el valor del PC seguirà la successió incremental (autoincrement en 4 a cada cicle) i apuntarà no a la posició ($0x20$) de la següent instrucció (*auipc x10 0x10000*) ja que s'està descodificant, sinó a la posició de dos instruccions més enllà ($0x28$), la qual és posició de la instrucció *sw x13 0(x10)*.

8 Qüestió 8

Quan es produeix el salt, quantes etapes de la instrucció "auipc x10 0x 10000" s'executen al *pipeline*?
Quan es produeix el salt només s'executen les dues primeres etapes (*Fetch i Decode*) ja que un cop executat el salt, canvia el valor del PC respecte del produït per l'autoincrement (nominal) i, per tant, es deixa de seguir executant les etapes d'aquesta instrucció perquè no s'ha d'executar segons el flux del programa afectat pel salt condicional que s'ha produït.

9 Qüestió 9

Quan s'està executant la instrucció de salt (etapa EX), i el salt es produeix, a quina posició apunta el program counter (PC)?
Si el salt es produeix, la posició a la qual apunta el PC és l'adreça de salt especificada. Com aquesta és <loop> ($0x14$), aquesta serà la posició a on apunti el PC un cop realitzat el salt condicional.

3 CONCLUSIONS

Aquesta pràctica, tot i ser més curta que les anteriors en nombre d'exercicis, sessions i temps de desenvolupament, m'ha permès entendre d'una manera més tècnica com s'implementen els salts condicionals al processador R5SP. A més a més he pogut comprovar i analitzar gràcies al simulador RIPES com les instruccions es divideixen en 5 microinstruccions, com es realitza el *pipeline* i com es reajusta quan es produeixen salts en les instruccions.

Finalment he aprés la diferència entre el RSCP i R5SP, al primer no es diferencia entre instrucció i microinstrucció ja que totes les instruccions són microinstruccions, en canvi al segon cada instrucció es divideix en 5. D'aquesta manera aconseguim reduir el cicle del rellotge, millorant la velocitat d'execució (usant tots els recursos diferenciats de la CPU simultàniament). Tanmateix, hi ha cops que això no és possible, per exemple hem vist que s'executen etapes de instruccions que després, per culpa dels salts condicionals, no serveixen ja que el flux del programa canvia i, per tant, el processador introdueix cicles d'espera per reajustar aquest *pipeline*.