Pràctica 4

Javier Pedragosa

INTRODUCCIÓ

1) Abans d'executar els codis tracta d'esbrinar la seva funcionalitat. Faran el mateix? Creus que 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.

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

Les etapes IF i ID es repeteixen perquè el processador preveu que la condició de salt no es compleixi així que continua amb normalitat, però quan la instrucció de salt condicional arriba a la etapa d'execució el processador interpreta que al final si que s'ha complert la condició i executa el salt, i descarta el que havien fet les etapes anteriors. En aquest cas és innecessari, ja que el salt és cap a les instruccions que el processador ha començat a executar amb antelació, però és el comportament esperat en un processador de 5 etapes.

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.

PRÀCTICA

Informe

Objectius

L'objectiu d'aquesta pràctica és la introducció a la execució en pipeline i tot el que comporta, com ara els conceptes de flush i stall.

Realització

En executar el codi 4, passada la línia de la instrucció de salt condicional, observem que es produeix un flush. Aquest flush és causat pel fet que el processador preveu que la condició no es compleix i continua la execució del codi sense fer el salt. A la primera iteració, però, la condició sí que es compleix i s'efectua el salt, la qual cosa provoca que els dos cicles que ha emprat en executar les últimes dues línies s'hagin de descartar.

En executar el codi 5, observem el mateix flush del codi 4, però a més, al final de la execució, observem també un stall. Aquest stall és provocat pel fet que volem sumar un immediat al contingut d'a0, però a0 està sent modificat per la línia anterior (carregant-li el valor que conté l'espai en memòria que hem anomenat guardaResultat).

Com que necessitem saber el valor d'a0 a la etapa de decodificació per poder executar bé la suma, hem d'esperar que la instrucció que ens carrega el valor en memòria a a0 passi per la etapa d'execució, per així poder accedir al veritable valor d'a0 en el moment que executem aquesta suma.

Conclusió

Gràcies a aquesta pràctica he aconseguit aprofundir en el funcionament d'un processador de 5 cicles, així com tot el que implica l'intent d'optimitzar un microprograma que s'executa en pipeline, com ara el fet d'assumir que una comparació a un salt condicional no es complirà mai, cosa que provoca estats de flush, a més del stall, que es produeix quan volem guardar un valor a memòria que encara no està calculat.

Preguntes sobre el simulador RIPES

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

- Codi 3:
  - Cicle 6:
    - Fetch: la a0, resultat
    - Decode: bgt a7, zero, salta
    - Execution: addi a7, a7, -1
    - Memory: add a3, a2, a3
    - Write-back: addi a2, zero, 4
  - Cicle 8:
    - Fetch: sw a3, 0(a0)
    - Decode: la a0, resultat (la segona part, ja que donada la quantitat d'adreçes de memòria, els adreçaments s'han de fer en dos cicles)
    - Execution: la a0, resultat (la primera part)
    - Memory: bgt a7, zero, salta
    - Write-back: addi a7, a7, -1
- Codi 5:
  - Cicle 6:
    - Fetch: addi a7, a7, -1
    - Decode: add a3, a2, a3
    - Execution: add a3, zero, zero
    - Memory: addi a2, zero, 9
    - Write-back: lw a7, 0(a0)
  - Cicle 8:
    - Fetch: la a0, guardaResultat
    - Decode: bgt a7, zero, loop
    - Execution: addi a7, a7, -1
    - Memory: add a3, a2, a3
    - Write-back: 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:
  - S'activa el Write Enable al banc de registres, i correspon a la instrucció add a3, a2, a3.
- Codi 5:
  - S'activa el Write Enable al banc de registres, i correspon a la instrucció addi a2, zero, 9.

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

- Codi 3:
  - 268435480 i -24
- Codi 5:
  - 28 i -8

4) Què està calculant la ALU al cicle 9?

- Codi 3:
  - Està calculant la adreça de resultat, que com que tenim més adreces de memòria de les que podem expressar en el bus d'adreces s'ha de calcular en dos cicles. Concretament aqui està al segon cicle.
- Codi 5:
  - Està calculant la adreça de guardaResultat, que com que tenim més adreces de les que podem expressar en el bus d'adreces s'ha de calcular en dos cicles. Concretament aqui està al segon cicle.

(A partir d'aquest punt només analitzo el codi 5).

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:

Al cicle 10 es presenta un flush a causa que el processador fa una predicció respecte al resultat de la comparació que es fa al salt condicional, i assumeix que no es complirà i continua la execució com tal. El flush ve perquè aquesta predicció és incorrecta, la comparació sí que es compleix, llavors ha de fer el salt i el pipeline ha de tornar a començar la execució des de la primera fase del pipeline a la instrucció corresponent.

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

S'executen totes etapes, les 5.

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 36.

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

S'executen dues, el fetch i el decode.

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

El PC apunta a la posició 20.