Számítógép Architektúrák

Pipeline feladatok (IMSc)

1. Feladat Legyen adott az alábbi utasítás sorozat:

```
i1:  R0 ← MEM [R1+8]
i2:  R2 ← R0 * 3
i3:  R3 ← MEM [R1+12]
i4:  R4 ← R3 * 5
i5:  R0 ← R2 + R4
```

Az utasítássorozatot egy, az órán tanult 5 fokozatú pipeline-t használó processzor hajtja végre.

(a) Azonosítsa az összes előforduló adat-egymásrahatást!

(b) Adja meg az utasítássorozat ütemezését (melyik utasítás mikor melyik fázisban van)! Ha szünetet kell beiktatni, jelezze, hogy mi az oka! Használja az alábbi jelöléseket:

- A^*: a szünet oka adategymásrahatás
- F^*: a szünet oka feldolgozási egymásrahatás
- P^*: a szünet oka procedurális egymásrahatás

(c) Minden egyes utasításnál jelölje, hogy amikor az EX fázisba ér, szükség van-e forwarding-ra! Ha igen, adja meg, hogy melyik pipeline regiszterből kell a szükséges értéket visszacsatolni!

(d) Ha egységnyi ideig tart minden fázis, mennyivel gyorsabb a pipeline-al való végrehajtás, mint anélkül?

(e) Rendezze át az utasítássorozatot úgy, hogy az a lehető leggyorsabban fusson le, gyorsabban, mint eredetileg!

2. Feladat Legyen adott az alábbi utasítás sorozat:

```
i1:  R5 ← MEM [R3+24]
i2:  R6 ← MEM [R4+16]
i3:  R7 ← R6 + R5
i4:  R8 ← R6 - R5
i5:  R5 ← R7 * R8
i6:  R4 ← R4 + 4
```

Az utasítássorozatot egy 6 fokozatú utasítás pipeline-t használó processzor hajtja végre. A pipeline minden utasítás végrehajtását 5 részműveletre bontja: betölti (IF), dekódolja (ID), végrehajtja a vonatkozó aritmetikai műveletet (EX), majd a vonatkozó memóriaműveletet (MEM), végül a regisztertárolóba írja az eredményregiszter értékét (WB). Az ID, EX, MEM és WB fázisok késleltetése 1 órajel, az IF fázis késleltetése 2 órajel, de iterációs ideje 1 órajel (vagyis 2 pipeline fokozatként jelenik meg: IF0 és IF1). Minden utasítás végrehajtása mind az 5 részműveleten átesik, függetlenül attól, hogy szüksége van-e rá. Minden forwarding út használata megengedett. Ha bármilyen egymásrahatás az utasítás megállítását igényli, az utasítás mindig a legutolsó olyan fázisban áll meg, ameddig egymásrahatás nélkül eljut.

(a) Azonosítsa az összes előforduló adat-egymásrahatást!

(b) Adja meg az utasítássorozat ütemezését (melyik utasítás mikor melyik fázisban van)! Ha szünetet kell beiktatni, jelezze, hogy mi az oka! Használja az A^*, F^*, P^* jelöléseket!

(c) Minden egyes utasításnál jelölje, hogy amikor az EX fázisba ér, szükség van-e forwarding-ra! Ha igen, adja meg, hogy melyik pipeline regiszterből kell a szükséges értéket visszacsatolni!

(d) Rendezze át az utasítássorozatot úgy, hogy az a lehető leggyorsabban fusson le, gyorsabban, mint eredetileg!

3. Feladat Legyen adott az alábbi utasítás sorozat:

```
i1:  D0 ← D1 * D2
i2:  D3 ← D0 + D5
i3:  MEM [R0+4] ← D3
i4:  MEM [R0+12] ← D0
```

A D0-D5 regiszterek lebegőpontos számokat, az R0 egész számot tárol.

A pipeline minden utasítás végrehajtását 5 részműveletre bontja: betölti (IF), dekódolja (ID), végrehajtja a vonatkozó aritmetikai műveleteket, majd a vonatkozó memóriaműveleteket (MEM), végül a regisztertárolóba írja az eredményregiszter értékét (WB). Minden utasítás mind az 5 részműveleten átesik, függetlenül attól, hogy szüksége van-e rá. Az IF, ID, MEM és WB fázisok késleltetése 1 órajel. Az aritmetikai művelet lehet egész művelet, mely késleltetése 1 órajel (EX, a címszámítást is ez az egység végzi), lebegőpontos összeadás, melynek késleltetése 4, iterációs ideje 1 (A0, A1, A2, A3), valamint lebegőpontos szorzás, melynek késleltetése 7, iterációs ideje pedig 1 (M0, M1, M2, M3, M4, M5, M6). A három aritmetikai egység képes párhuzamos működésre, és az utasítások soron kívüli befejezésére, ha ennek semmilyen szemantikai vagy egymásrahatásbeli akadálya nincs. Minden forwarding út használata megengedett. Ha bármilyen egymásrahatás az utasítás megállítását igényli, az utasítás mindig a legutolsó olyan fázisban áll meg, ameddig egymásrahatás nélkül eljut.

(a) Azonosítsa az összes előforduló adat-egymásrahatást!

(b) Adja meg az utasítássorozat ütemezését (melyik utasítás mikor melyik fázisban van)! Ha szünetet kell beiktatni, jelezze, hogy mi az oka! Használja az A^*, F^*, P^* jelöléseket!

(c) Rendezze át az utasítássorozatot úgy, hogy az a lehető leggyorsabban fusson le, gyorsabban, mint eredetileg!

(d) Adja meg az átrendezett utasítássorozat ütemezését is, a szünetek okának megjelölésével!

4. Feladat Az Intel első pipeline-al rendelkező processzora, a 80486 is 5 fokozatú futószalaggal rendelkezett, a fázisok azonban eltértek a tanult pipline fázisaitól. Az IF és az ID után egy AG (address generation) fázis következett, ami a memóriakezelő utasítások számára címet számolt, az aritmetikai utasítások esetében pedig nem csinált semmit. Az AG után a MEM/EX jön, ami memória load/store esetén elvégezte a tényleges memóriaműveletet (az AG által számolt címre), aritmetikai utasítások esetén pedig magát az aritmetikai műveletet. Az utolsó fázis itt is a WB, az eredmény visszaírása a regiszter tárolóba.

(a) Ütemezze az 1. feladat példaprogramját ezen az alternatív pipeline struktúrán! Melyik pipeline tudta gyorsabban végrehajtani?

(b) Mely utasításpárokat tudja hatékonyabban végrehajtani a tanult, és melyeket ez az alternatív pipeline? Melyik a jobb választás?

5. Feladat Egy 2-utas in-order szuperskalár processzor az órán megismert egyszerű, 5-fokozatú pipeline-al rendelkezik, melynek szélessége 2. A processzor ideális abból a szempontból, hogy bármely két egymást követő utasítás végrehajtását el tudja indítani, és tökéletes elágazásbecslővel rendelkezik. Az utasításpárokra csak annyi a korlátozás, hogy egyszerre kell átesniük az IF és ID fázisokon.

A fordító a

```
for (i=0; i!=j; i++)
    b[i] = a[i];
```

C program alapján az alábbi alacsony szintű utasítássorozatot állította elő:

```
i1:        R1 ← 0
i2:  iter: JUMP end IF R1==R2
i3:        R6 ← R3 + R1
i4:        R7 ← MEM[R6]
i5:        R8 ← R4 + R1
i6:        MEM[R8] ← R7
i7:        R1 ← R1 + 1
i8:        JUMP iter
i9: end:
```

(Figyeljük meg, hogy az `MEM[R4+R1]← R7` utasítás nem megengedett, hiszen három forrásoperandusa lenne. Az ortogonalitás miatt ekkor viszont `R7 ← MEM[R3+R1]` utasítás sem megengedett, ezért a címeket külön kell számolni aritmetikai utasításokkal.)

(a) Ütemezze az utasítássorozatot a 2-utas in-order pipeline-ra, a ciklus két iterációjáig!

(b) Rendezze át az utasítássorozatot úgy, hogy a program futása a lehető leggyorsabb legyen!