Pipeline feladatok (IMSc)

4. Feladat megoldása

(a) Megoldás:

R0 ← MEM [R1+8]:	IF	ID	AG	E/M	WB				
R2 ← R0 * 3:		IF	ID	AG	E/M	WB			
R3 ← MEM [R1+12]:			IF	ID	AG	E/M	WB		
R4 ← R3 * 5:				IF	ID	AG	E/M	WB	
R0 ← R2 + R4:					IF	ID	AG	E/M	WB

Tehát erre a programra gyorsabb az Intel pipeline-ja.

(b) A tanult pipeline jobb a `R0 ← R2 + R4, R1 ← MEM[R0]` jellegű (címet számolok és betöltöm) utasításpárokban (nincs késleltetés), az Intel pipeline itt egy késleltetés beiktatását igényli. Az Intel jobb a `R0 ← MEM [R1+8], R2 ← R0 * 3` párokban (betöltöm és használom). Melyik a jobb? Attól függ, milyen a workload, milyen alkalmazásra készül a processzor.

5. Feladat megoldása 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 (akkor is, ha adatfüggés van közöttük), 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:
```

(a) Az ütemezés:

i1:	IF	ID	EX	MEM	WB							
i2:	IF	ID	A^*	EX	MEM	WB						
i3:		IF	F^*	ID	EX	MEM	WB					
i4:		IF	F^*	ID	A^*	EX	MEM	WB				
i5:				IF	F^*	ID	EX	MEM	WB			
i6:				IF	F^*	ID	A^*	EX	MEM	WB		
i7:						IF	F^*	ID	EX	MEM	WB	
i8:						IF	F^*	ID	EX	MEM	WB	
i2:								IF	ID	EX	MEM	WB
i3:								IF	ID	EX	MEM	WB
i4:									IF	ID	EX	MEM	WB
i5:									IF	ID	EX	MEM	WB
i6:										IF	ID	EX	MEM	WB
i7:										IF	ID	EX	MEM	WB

Észrevehető, hogy minden páratlan iteráció rázós (függő utasítások kerülnek egy csoportba), és minden páros sima ügy lesz.

(b) Átrendezett kód (a trükk a szokásos):

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