MIPS Instruction'lar

Kodun:

```
assembly
KopyalaDüzenle
I1: ADDI $s0, $0, 10      # $s0 = 10
I2: ADDI $s1, $0, 5       # $s1 = 5
I3: SLL  $s0, $s0, 4      # $s0 = $s0 << 4 = 160
I4: LW   $s2, 0($s0)      # $s2 = Mem[160]
I5: ADD  $s2, $s2, $s1    # $s2 = $s2 + $s1 (RAW hazard from LW)
I6: SW   $s2, 4($s0)      # Mem[164] = $s2
```

Beklenen Hazard Durumu

RAW Data Hazard:

- **I4 → I5 arasında** var, çünkü:
  - I4 bir `LW` (load), sonucu `$s2`'ye yazar.
  - I5 hemen ardından `$s2`'yi okuyor.
  - `LW` sonucu ancak **MEM aşamasında** elde edilir.
  - Yani `I5`'i **1 clock cycle** stall etmeliyiz.

Control Hazard (bu örnekte yok ama genelde olur):

- Jump ya da Branch olsaydı, iki komutu bubble yapmamız gerekirdi.

Clock Cycle Pipeline Tablosu

Clock	IF	ID	EX	MEM	WB	Açıklama
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	**I5 ID aşamasında ama $s2 hazard**
6	NOP	I5	I4	I3	I2	⛔ Stall! I5 ilerleyemez → Bubble
7	I6	NOP	I5	I4	I3	I5 artık hazard yok → ilerler
8		I6	NOP	I5	I4	
9			I6	NOP	I5	
10				I6	NOP	
11					I6	

Beklenen Register Değerleri

Register	Beklenen Değer	Clock'tan Sonra Değişir
`$s0`	160	I3 (Shift) sonrası
`$s1`	5	I2 sonrası
`$s2`	Mem[160]+5	I5 sonrası

Hafıza Değişiklikleri

Address	Beklenen Değer	Açıklama
160	(input verilmiş olmalı)	I4 okur
164	`$s2` = Mem[160] + 5	I6 yazar