CS 3725 Feb. 14/12

<u>Dependency</u> can cause inst. issuing bottleneck

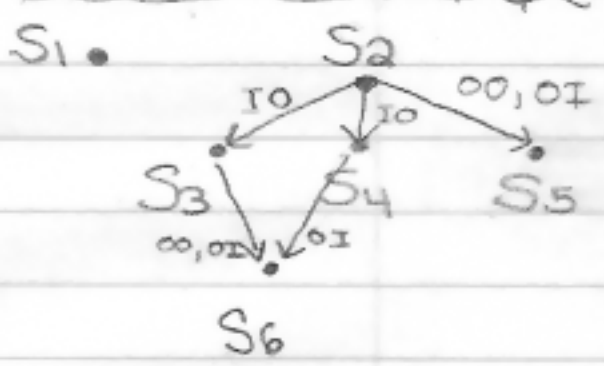
- D-Unit - can be alleviated by virtual d-units
- Data Dependency
  - -IO } can be alleviated by renaming registers
  - -OO }
  - -OI → ① Scoreboard + Conditional Issue of Instruction
    ② How MIPS deals with OI (Instruction Forwarding)

<u>Recap</u> Register Renaming

ex: (earlier example)

S_1: $R_6 \leftarrow R_6 * R_6$
S_2: $R_1 \leftarrow R_2 + R_3$
S_3: $R_2 \leftarrow R_4 * R_5$
S_4: $R_3 \leftarrow R_3 + R_4$
S_5: $R_1 \leftarrow R_1 + R_5$
S_6: $R_2 \leftarrow R_3 + R_4$

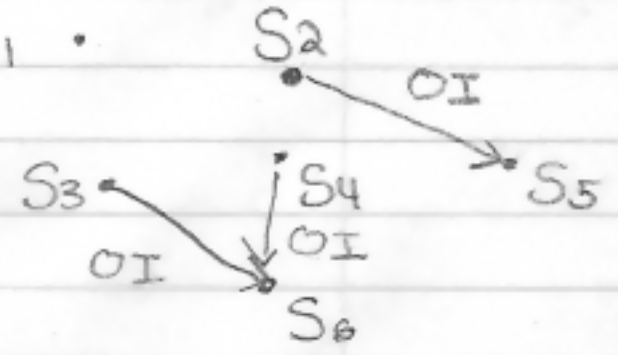
<u>Precidence Graph</u>



<u>Reg. Renaming</u>

S_1: $R_6^{0b} \leftarrow R_6^{a} * R_6^{a}$
S_2: $R_1^{a} \leftarrow R_2^{a} + R_3^{a}$
S_3: $R_2^{b} \leftarrow R_4^{a} * R_5^{a}$
S_4: $R_3^{b} \leftarrow R_3^{a} + R_4^{a}$
S_5: $R_1^{b} \leftarrow R_1^{a} + R_5^{a}$
S_6: $R_2^{c} \leftarrow R_3^{b} + R_4^{a}$

<u>Precidence Graph after Renaming</u>

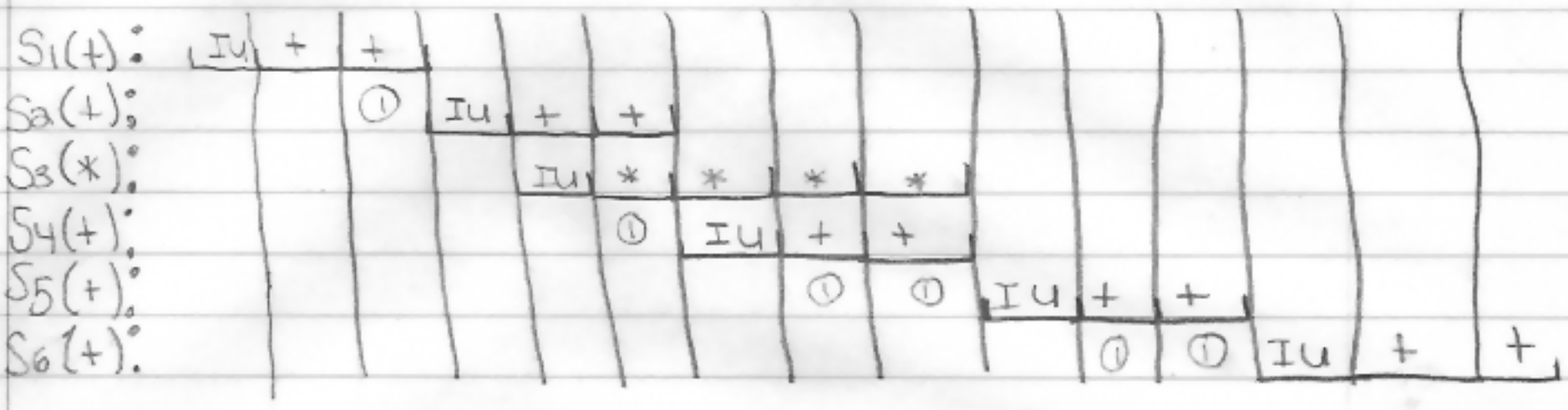


<u>Terminology</u>

IO - Antidependency
OI - True Dependency

Ex: Given P (after renaming)

The Machine (M): 1 +'er + time = 2
1 *'er * time = 4
I-U time = 1



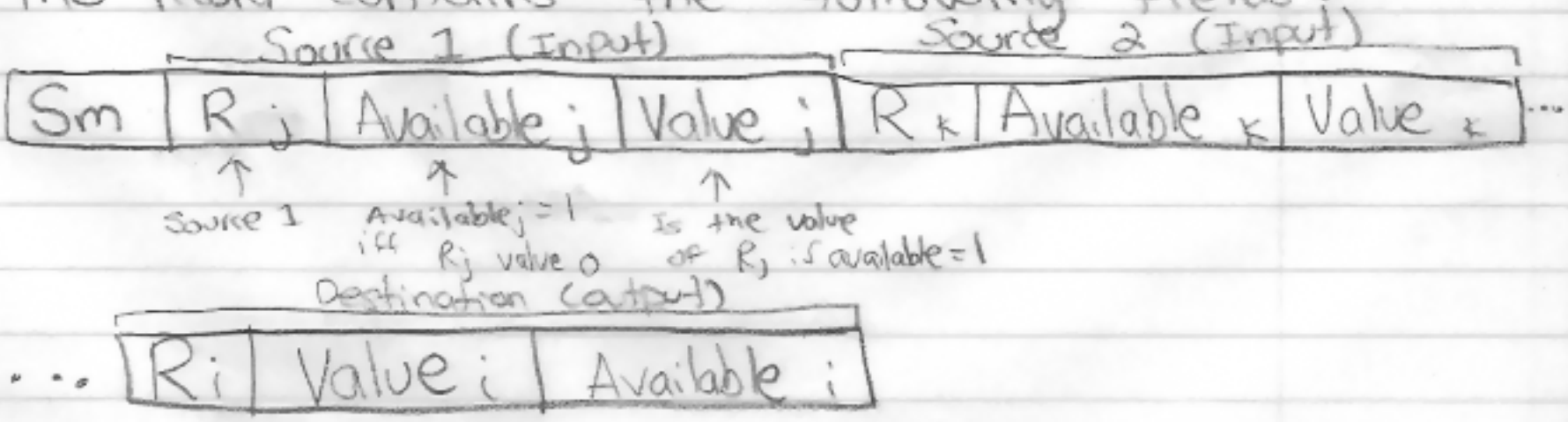
① Due to + dependency

How to Alleviate Bottleneck due to OI Dependency

Use a Score Board to Conditionally Issue an Instruction ↳(Reservation Table)

Idea of Conditional Issue of Instruction

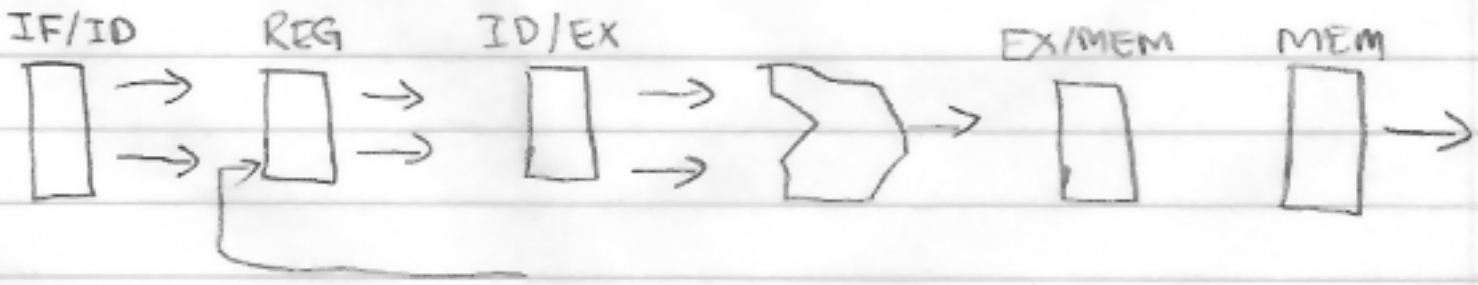
- An instruction is issued (conditionally to the scoreboard)
- The instruction is dispatched as a record
- The record contains the following fields:



- In the scoreboard we maintain one entry per D-unit
- A D-unit can execute an instruction if both of its source available flags are 1
- After processing the results, it searching the input fields in all the entries in the scoreboard and updates the relevant fields as necessary

How does MIPS Handle Dependencies?

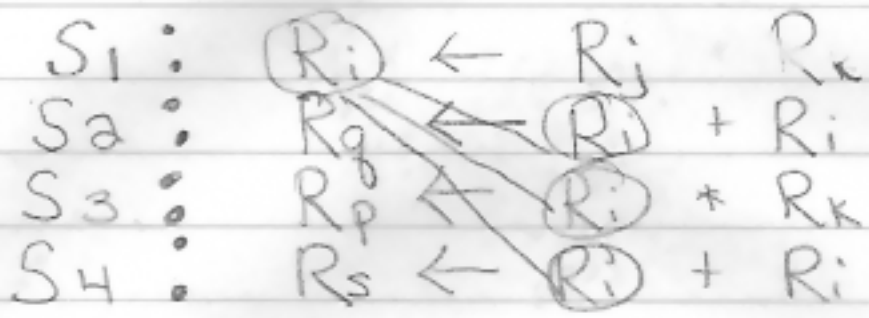
- One D-Unit (and hence no D-unit dependency)
- O-O ? x
- I-O ? x
- O-I



a) Output is produced either by
   (i) ALU (R-Type)
   (ii) MEM (lw)

Consider only R-type instructions

Ex:



Note: If the "distance" between two dependent instruction is more than 2, then the following inst will get the correct value.

MIPS uses "Instruction Forwarding"

If there is OI dependency between two consecutive instructions then MIPS circulates the results back to the inputs of the ALU with the correct value.