CS 3725 Feb 7/12

Data Dependencies
(aka Procedural Dependency or Register Dependency)

Key factors for correctly running S_i and S_j (instructions) in parallel.

① They do not write into the same register.

② The input register(s) of one instruction do not overlap with the output of another instruction.

Berstein's Conditions of Detecting + Exploiting Data Dependencies (Data Parallelism)

- Consider two instructions (S_i and S_j)
  $S_i : I_i \rightarrow O_i$ and $S_j : I_j \rightarrow O_j$
  where $i < j$ (i is the "previous instruction" and j is the "subsequent instruction")

[1] IO Dependency - Input [of the previous instruction] to Output [of the subsequent instruction]

$I_i \cap O_j = \{\emptyset\}$ iff S_i and S_j do not have IO Dependency

2. OI dependency (Output [of the prev instruction] to Input [of the subsequent instruction] dependency

$O_i \cap I_j = \{\emptyset\}$ iff S_i and S_j do not have OI dependency

3. OO dependency – (Output [of the prev inst] to Output [of the subsequent inst])

$O_i \cap O_j = \{\emptyset\}$ iff S_i and S_j do not have OO dependency

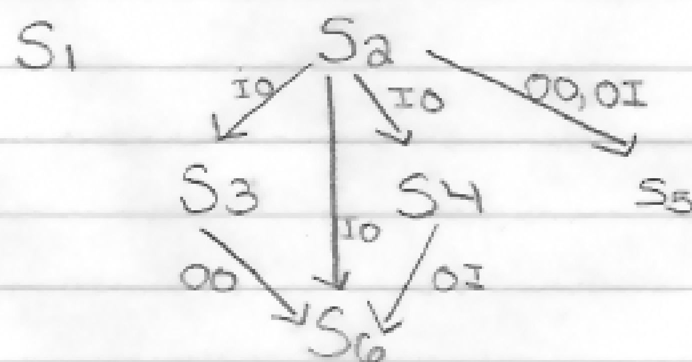
Data Dependency Graph

Gives a directed graph showing data dependencies among pairs of instructions.

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: $R_2 \leftarrow R_3 + R_4$

} Program Segment "p"



① The graph tells you about <u>inherent parallelism</u> (logical parallel) within a program.

② It allows for <u>Instruction Rescheduling</u>

③ <u>Real Parallelism</u> (no. of independent parallel units that the machine has)

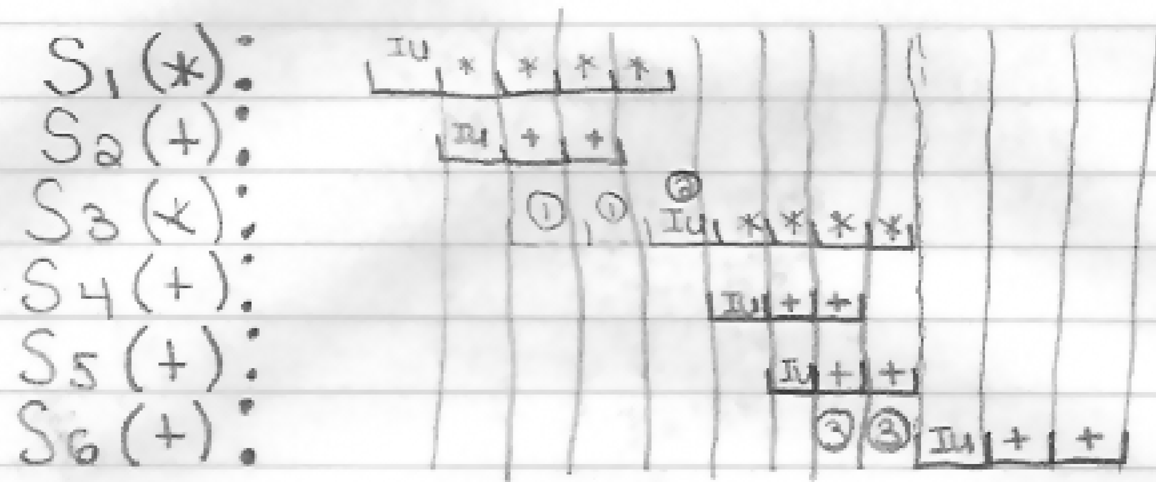
④ Can also be done at the execution time

⑤ Use of the <u>Transitive Closure Property</u> for Simplifying Procedural Graphs

eg. $S_a \rightarrow S_4 \rightarrow S_6$ covers $S_a \rightarrow S_6$
Hence the edge $S_a \rightarrow S_6$ can be removed w/o any loss of dependency information.

⑥ In a bigger context the graph is generated on a "chunk" of code at a time.

<u>Ex:</u> Given the last program segment (P) and machine M_1:
- infinite number of D units (ie. No D-unit conflict)
  + time: 2   * time: 4
  IU time: 1



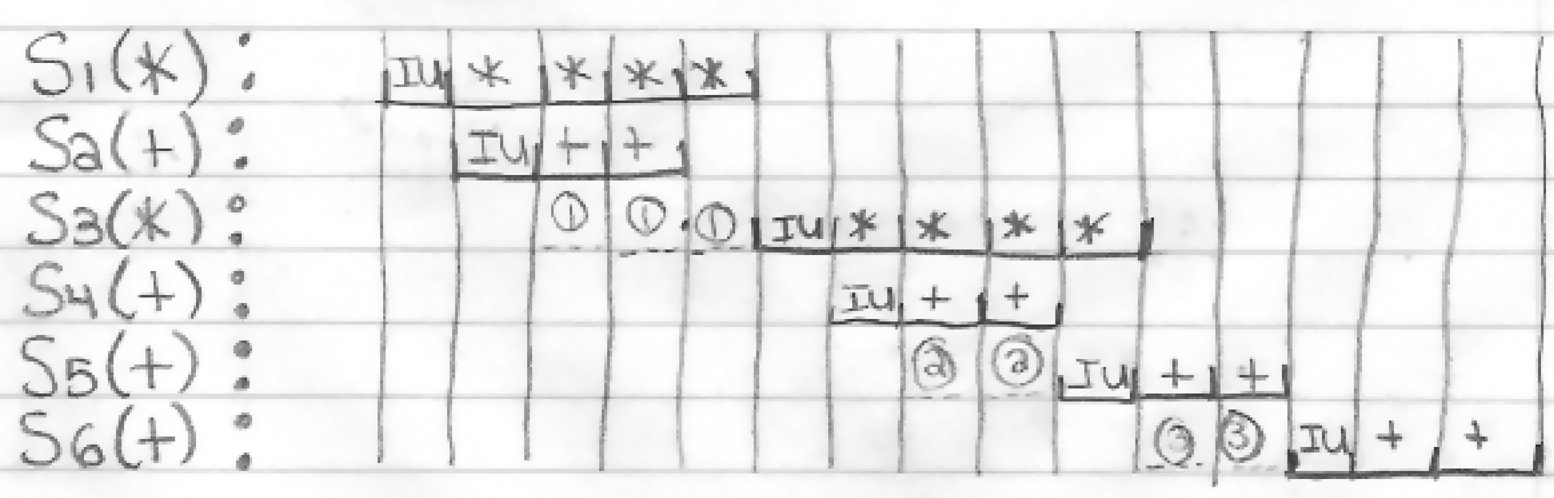
Annotations:

① S3 can not be issued at this time because S3 has a data dependency with S2.

② S3 can be issued because it does not have dependency with S1

③ S6 has a data dependency with S3

Ex2: Given the last program segment P and machine M1: - 1 adder, time = 2
- 1 multiplyer, time = 4



Annotations:

① S3 has * dependency with S1
② S5 has + dependency with S4
③ S6 has + dependency with S5