CS3725 Feb. 2/12

Detection and Exploitation of Parallelism

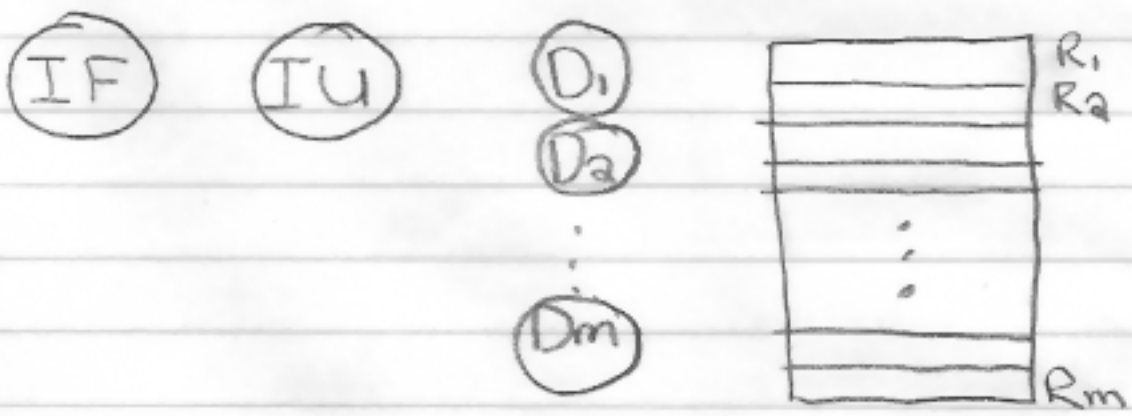
The (Generic) Computation Model

1. (Multi-functional) m D-units
   $D_1, D_2, \ldots, D_m$

2. n Global Registers
   $R_1, R_2, \ldots, R_n$

3. Two types of Instructions:
   ① $R_i \leftarrow D_n(R_j, R_k)$
   ② If $P(R_i)$ then Branch To...
   else ...

- Assume that the machine has
  - instruction prefetch
  - operand preload



⊛ The machine is:
In-order issue, out-of-order execution
(i.e. The instructions are issued one at a time and in the order which they appear in the program. Once the instructions are issued to the D-unit, then they may finish out-of-order).

Other considerations:

- Since the machine has instruction prefetch, we assume that IF does not take any time

- Since it has operand preload, we assume that data resides in the global registrars... $R_1, \ldots, R_n$

Def'n An instruction can only be issued if it can be executed (in ① the required D-unit is available ② and the registers are available)

(*) If an instruction cannot be issued, then subsequent instructions cannot be issued.

We study 4 types of dependency:
① D-unit dependency
② Data dependency
③ Order dependency
④ Control dependency

Some of the benefits of in-order issue and out of order execution:
- multiple inst running at the same time

- slower inst do not dictate the overall specs of the machine

But • we need to maintain the correctness of the program

Implication of D-Unit Dependency

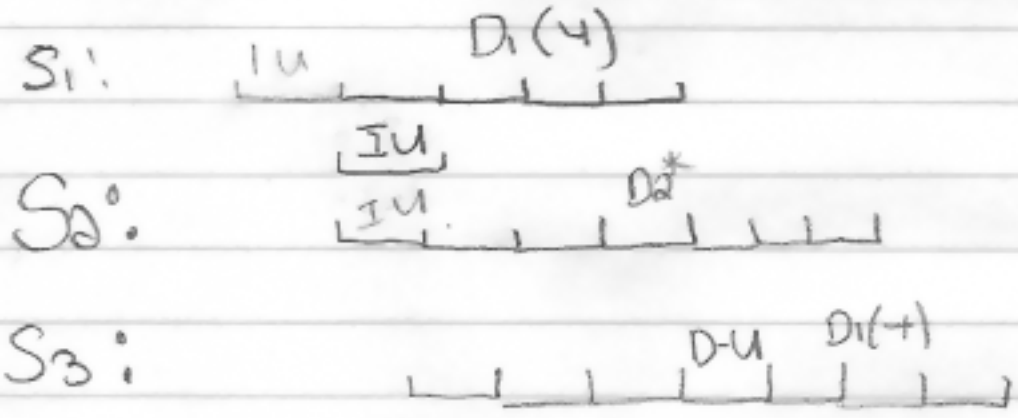
Ex: Consider a machine such that:

M: 1 Adder $D_1(+)$ + time = 4
1 Multiplier $D_2(*)$ * time = 7

Given Program segment:

P: S1 $R_1 \leftarrow R_2 + R_3$
S2 $R_4 \leftarrow R_2 + R_5$
S3 $R_6 \leftarrow R_3 + R_6$ I-U time = 1

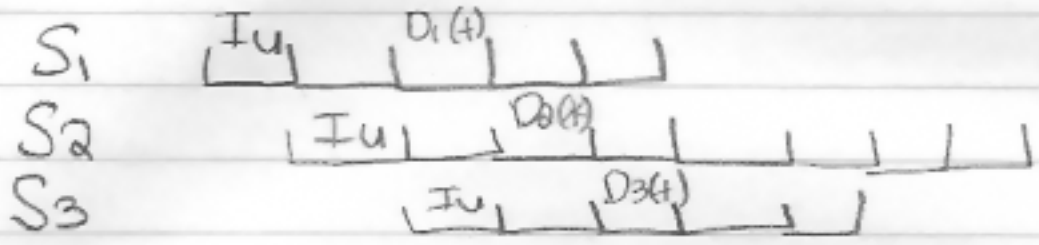
P-H Style Timing Diagram



① S_1 is ready to be issued
② S_2 can be issued because $D_2(6)$ B overlaps.
③ S_3 cannot be issued until $D_1(+)$ is available.

Total time to execute P in machine $M_1 = 10$.

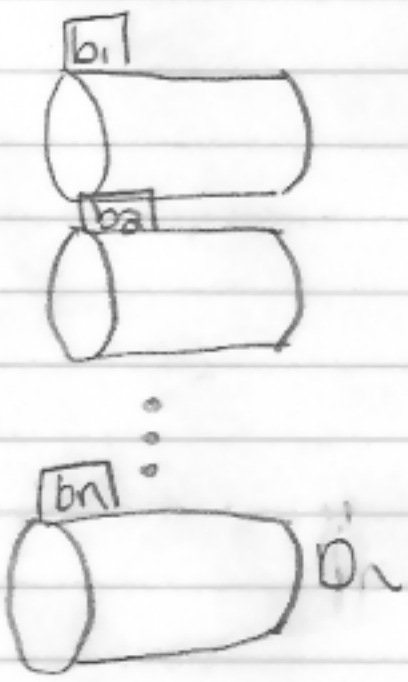
Ex2: M_2 same as M_1 except that M_2 has 2 adders $D_1(+)$, $D_2(*), D_3(+)$



(iv) S_3 can be issued to $D_3(+)$ even though $D_1(+)$ is busy.

How to detect D-unit availability at the Hardware Level

- We maintain a flip-flop (a flag) with each D-Unit
  - D_h
  - b_h



The way we use the FF D_n is busy if $B_n = 1$, D_n is available if $b_n \neq 0$.

Stages of Instruction Processing involving ONLY D-Unit Dependency

For instruction S : $R_i \leftarrow D_n(R_j, R_k)$

① Test b_n (Done in the I-U stage before issuing an instruction)

If $b_n = 1$, then "hold" issuing instruction S' otherwise set b_n

② Set b_n also done at the IU stage, before issuing S

$b_n \leftarrow 1$ (ie. S reserves D_n)

③ Issue instruction S to D_n (assume that there is no data dependency)

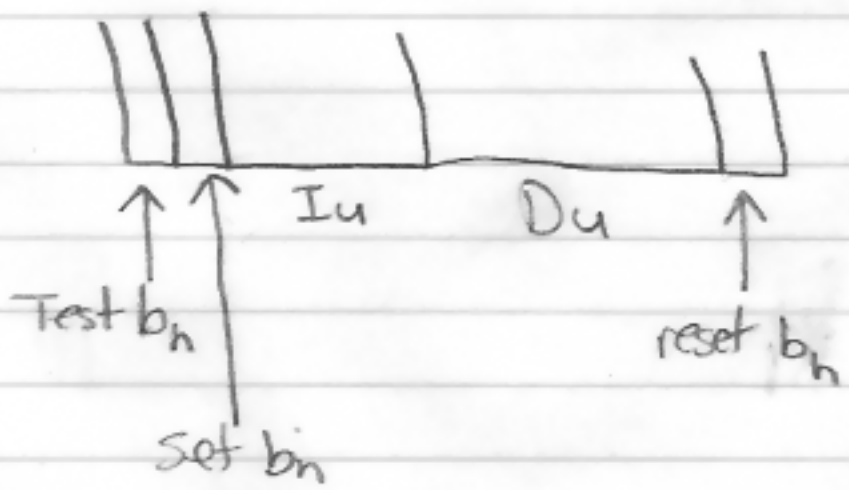
④ Execute S (in D_n)

⑤ Reset b_n

$b_n \leftarrow 0$

Releases D_n.

Timing

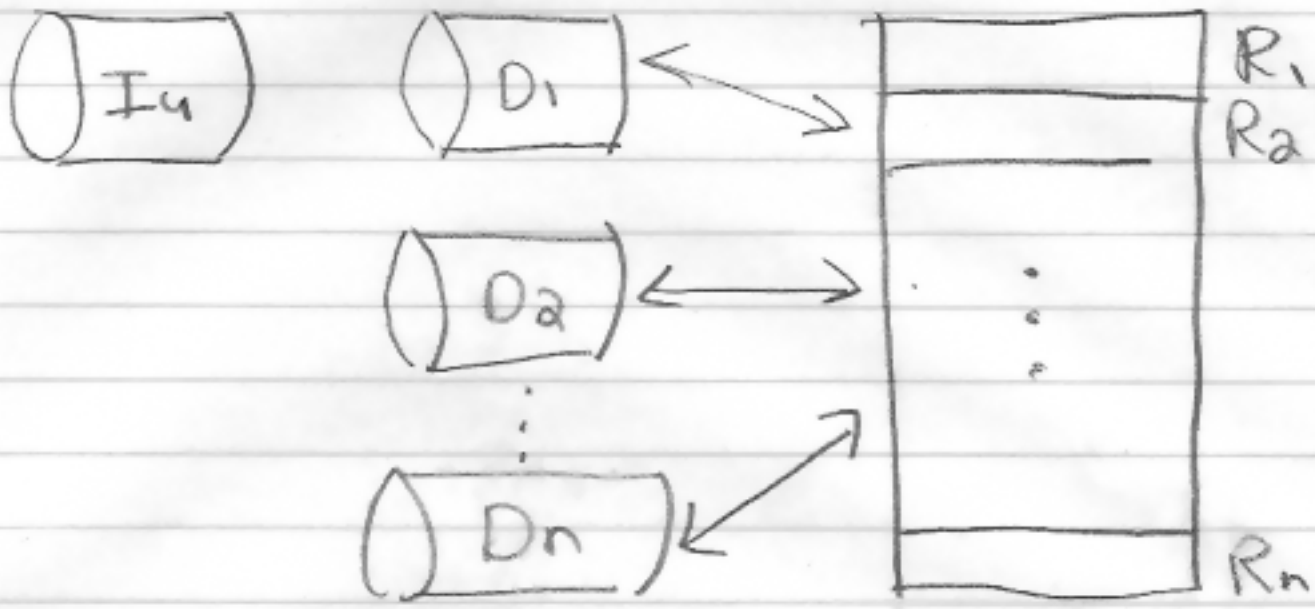


Q1: Implement this stages in software

Q2: Implement those stages at the hardware.

Data Dependency

(aka Register Dependency, Procedural Dependency)



Si and Sj can run in parallel iff
① Input to output must be disjoint
② Output to output must be disjoint

Bernstein's