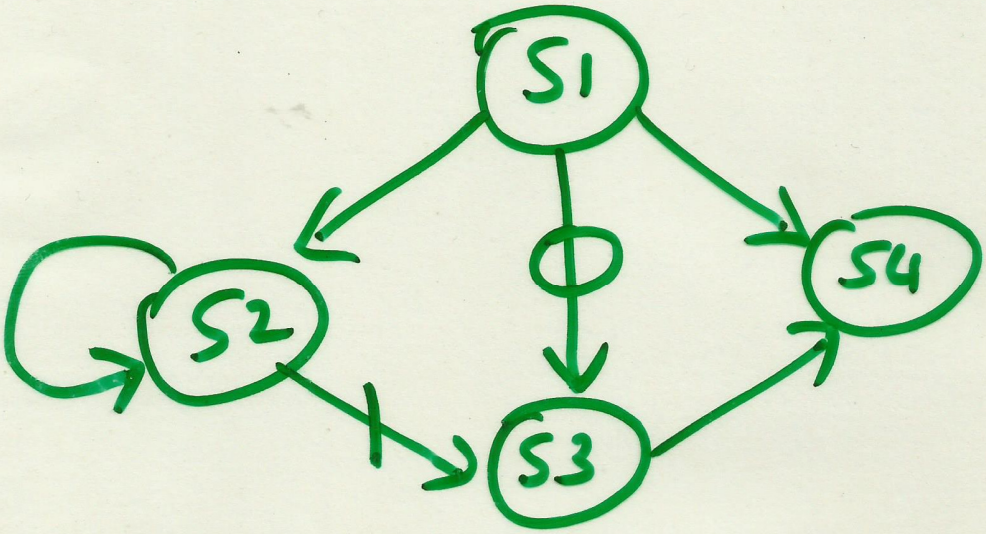
S1: Load R1, A / R1 ← mem(A)

S2: Add R2, R1 / R2 ← R1 + R2

S3: Mov R1, R3 / R1 ← R3

S4: Store B, R1 / mem(B) ← R1



- S2 (FD) on S1
- S3 (AD) on S2
- S3 (OD) on S1
- S2 & S4 are ||

- S2 is FD on itself

Example 2.2 . – page 55

P_1 : $C = D \times E$

P_2 : $M = G + C$

P_3 : $A = B + C$

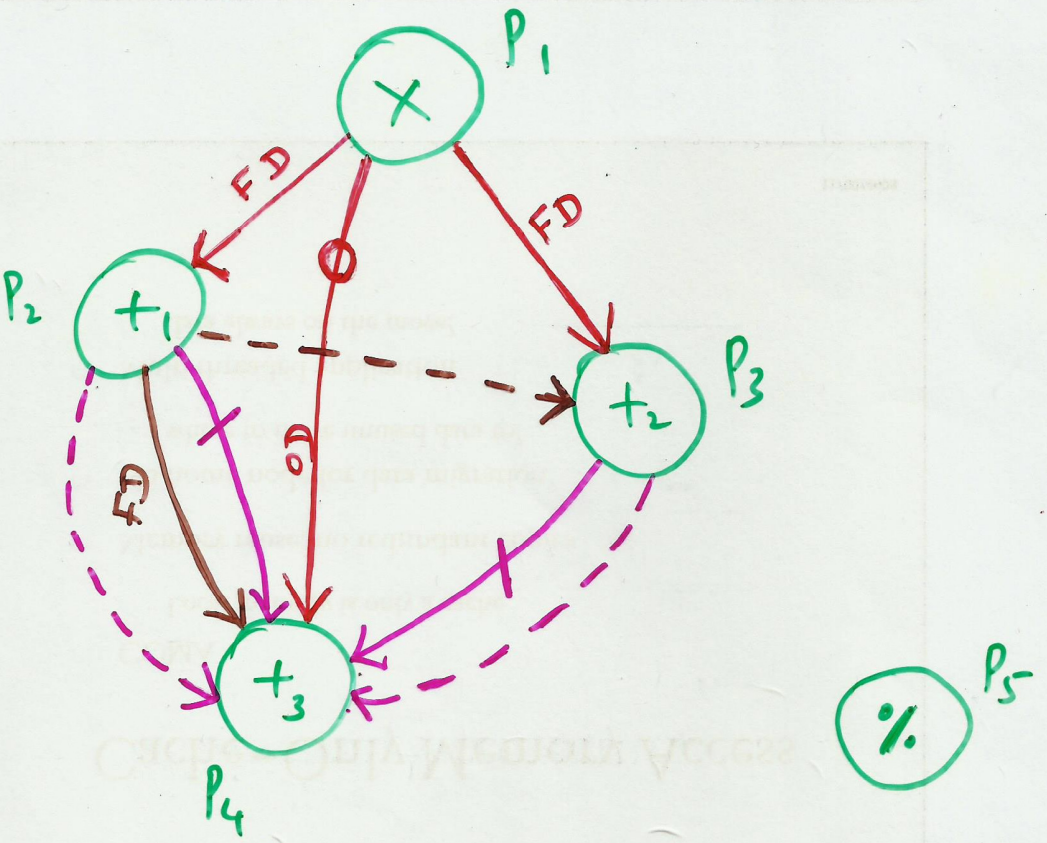
P_4 : $C = L + M$

P_5 : $F = G \div E$

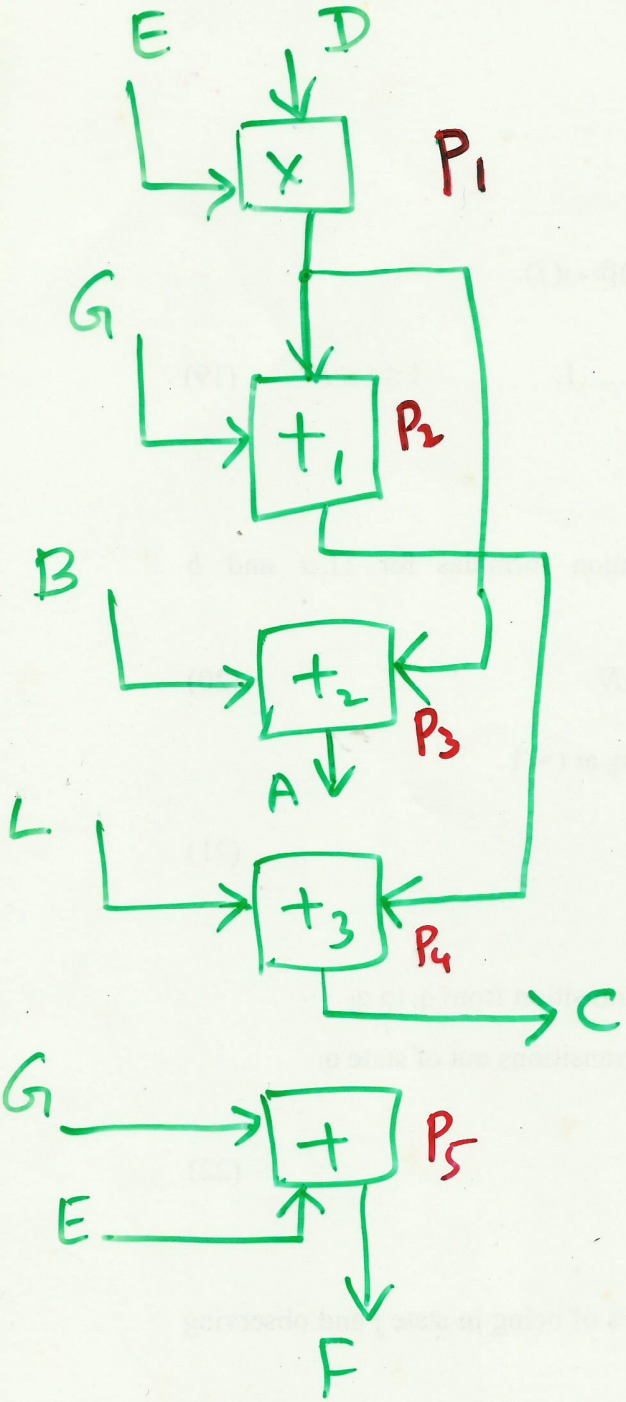
Assume:

- each statement needs one step to execute
- No pipelining

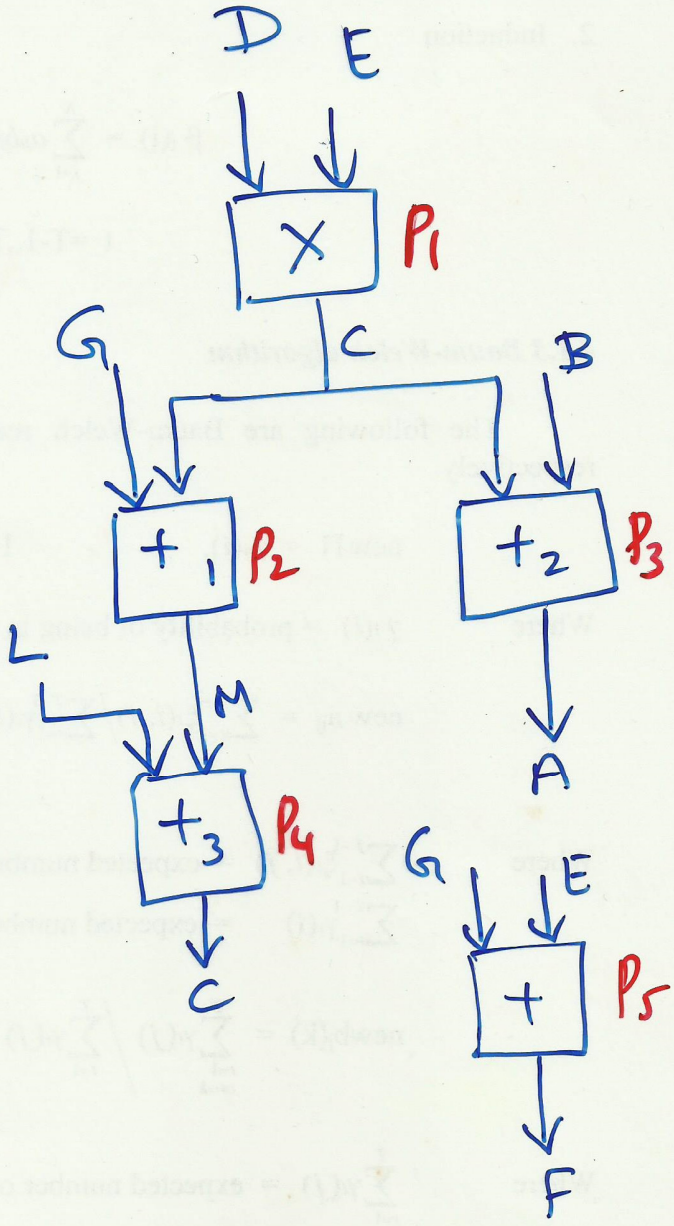
Dependence graph



Note: $I_2 \cap I_5 \neq \phi$ → does not affect exploiting the parallelism

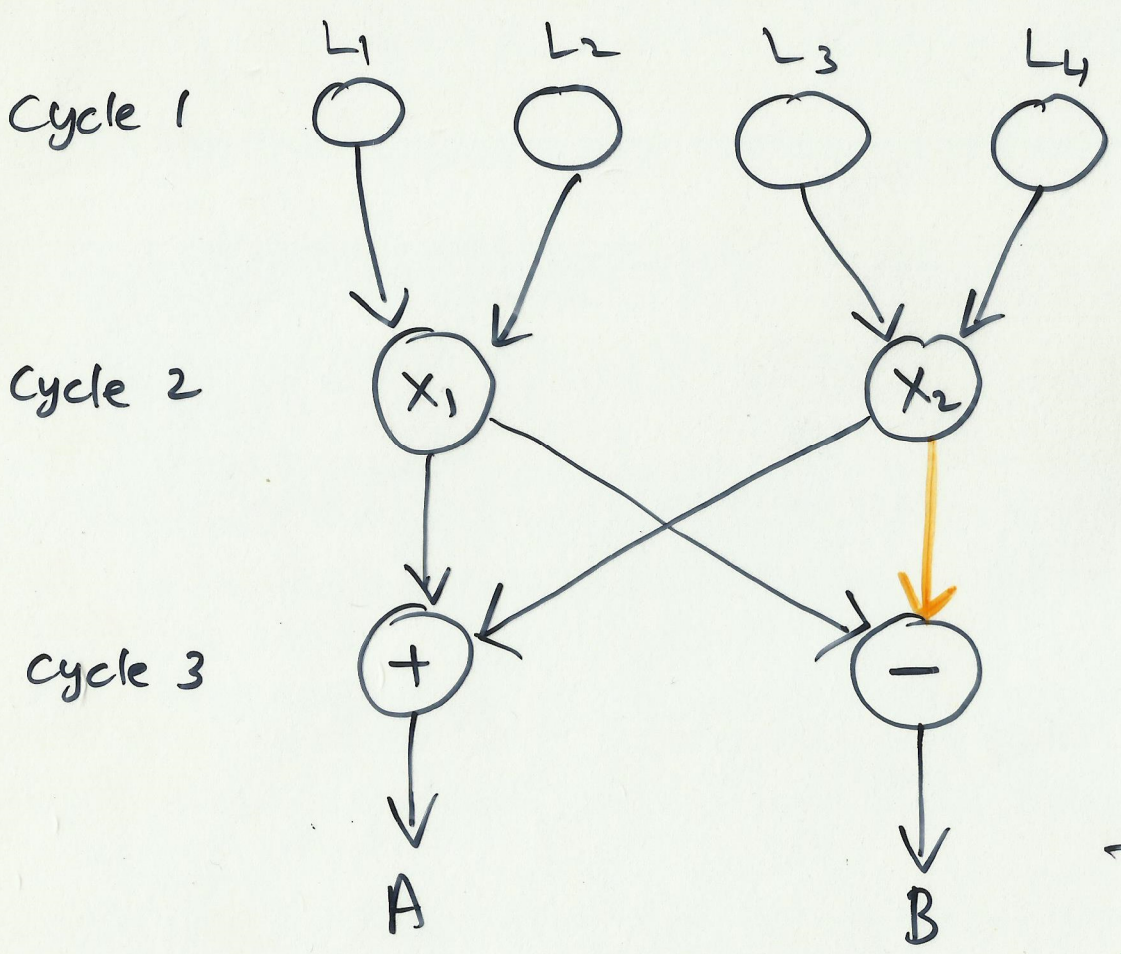


Seavential execution in 5 steps



Parallel execution (assuming 2 adders are available per step) simultaneously

Example 2.3 – page 58



- # of instructions per cycle

  8/3 = 2.67

Software Parallelism

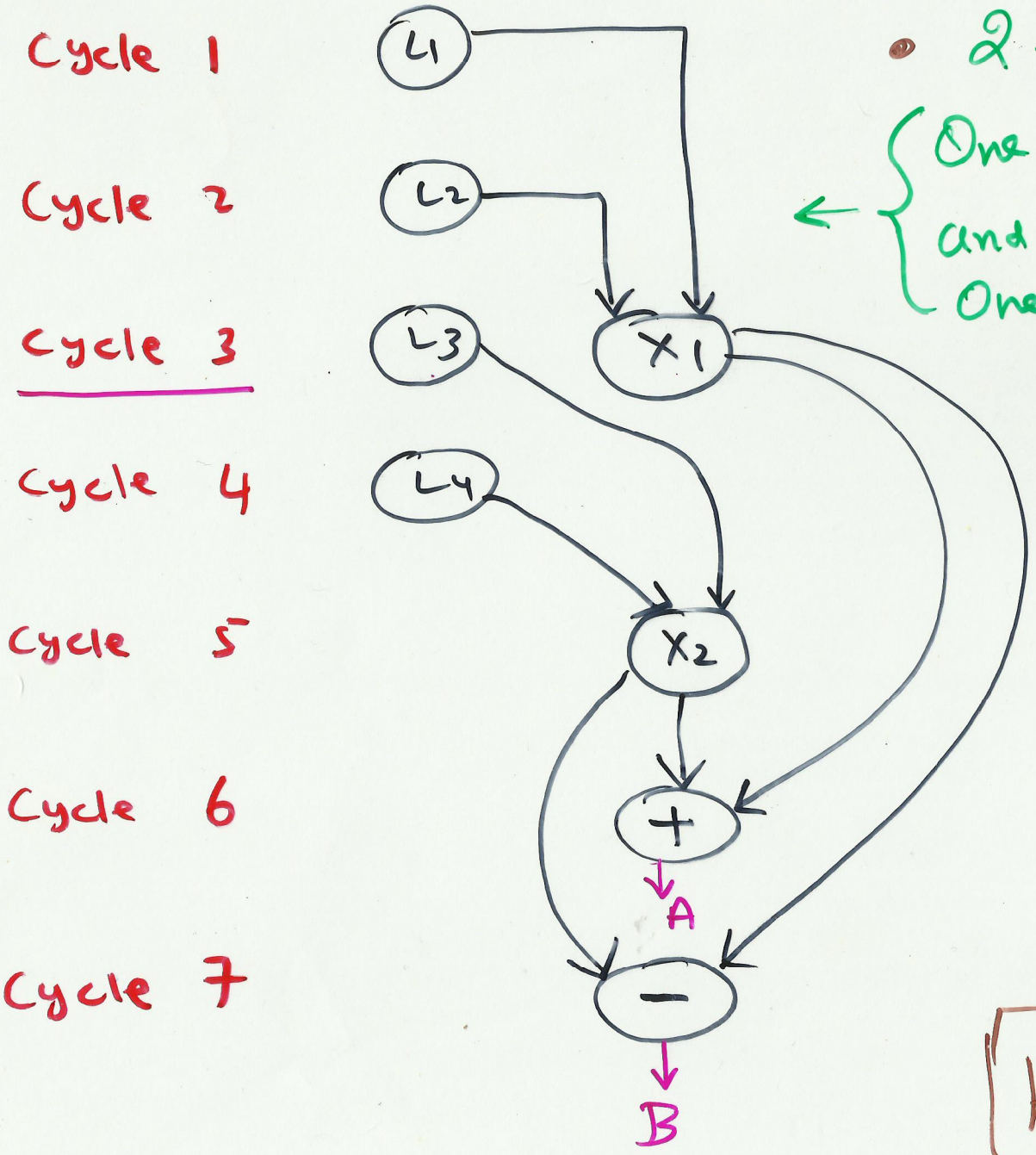
This is a program graph.

Li: Load operations → 4

arithmetic operations → 4

# of instructions 8

Example 2.3 (cont'd)



- 2-issue processor is given

← { One load/write [one memory access]
and
One arithmetic (+, ×, ÷, −)
simultaneously.

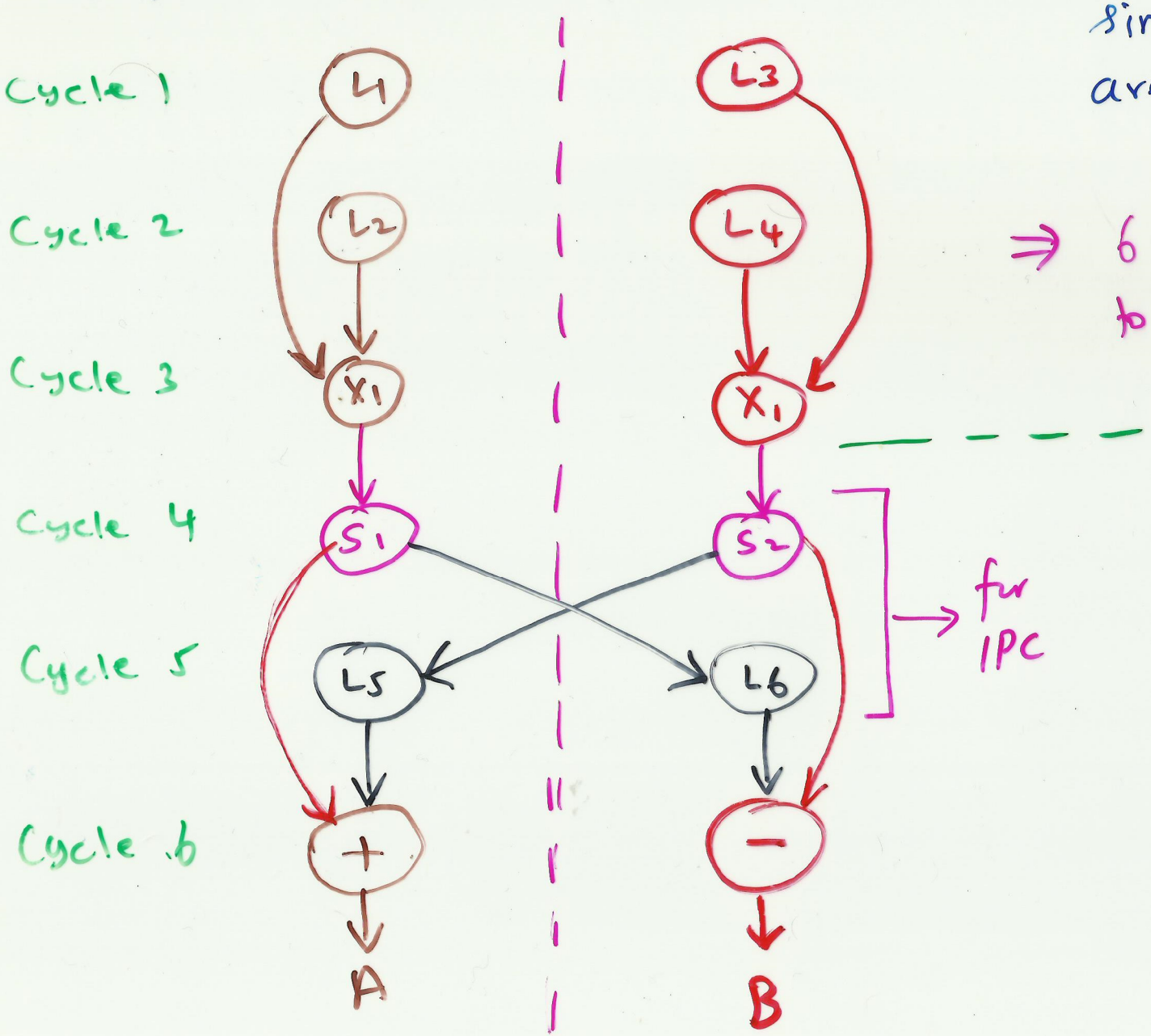
⇒ 8/7 = 1.14 instructions per cycle.

mismatch!

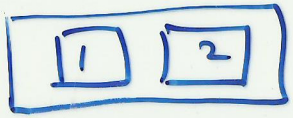
HARDWARE PARALLELISM

Example 2.3 Cont'd

Attempting to match the SP !!



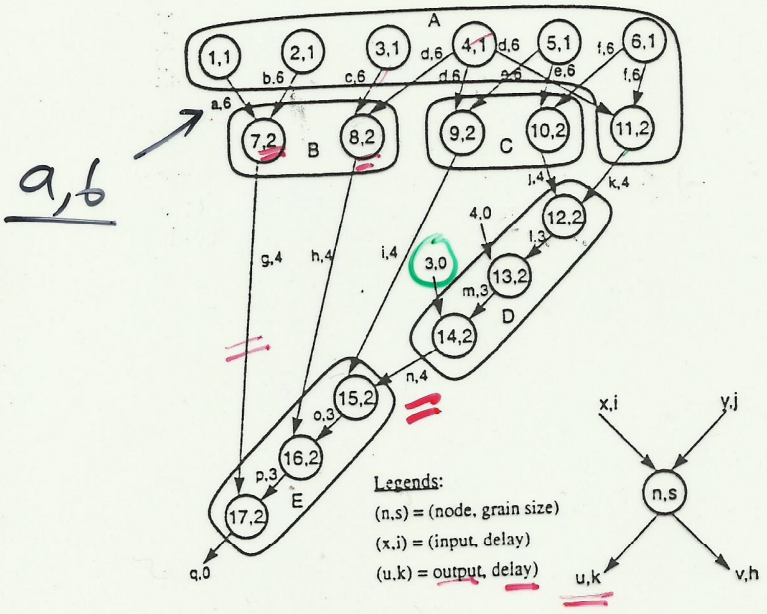
- dual-processors, but single-issue processors are given



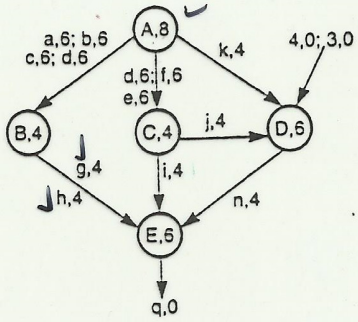
⇒ 6 cycles are needed to execute 12 instructions.

for IPC

s: # of machine cycles needed to execute all the operations within a node



(a) Fine-grain program graph before packing



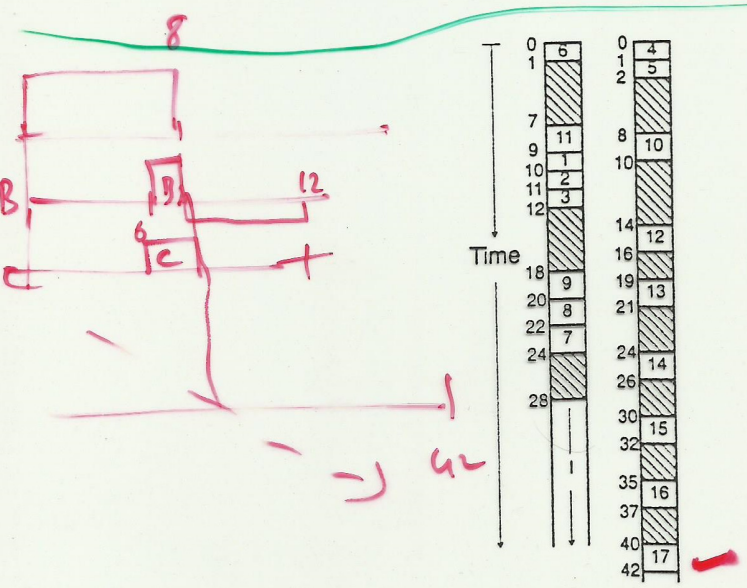
(b) Coarse-grain program graph after packing

Figure 2.6 A program graph before and after grain packing in Example 2.4. (Modified from Kruatrachue and Lewis, *IEEE Software*, Jan. 1988)

Begin

1.	$a := 1$	10.	$j := e \times f$
2.	$b := 2$	11.	$k := d \times f$
3.	$c := 3$	12.	$l := j \times k$
4.	$d := 4$	13.	$m := 4 \times l$
5.	$e := 5$	14.	$n := 3 \times m$
6.	$f := 6$	15.	$o := n \times i$
7.	$g := a \times b$	16.	$p := o \times h$
8.	$h := c \times d$	17.	$q := p \times q$
9.	$i := d \times e$		

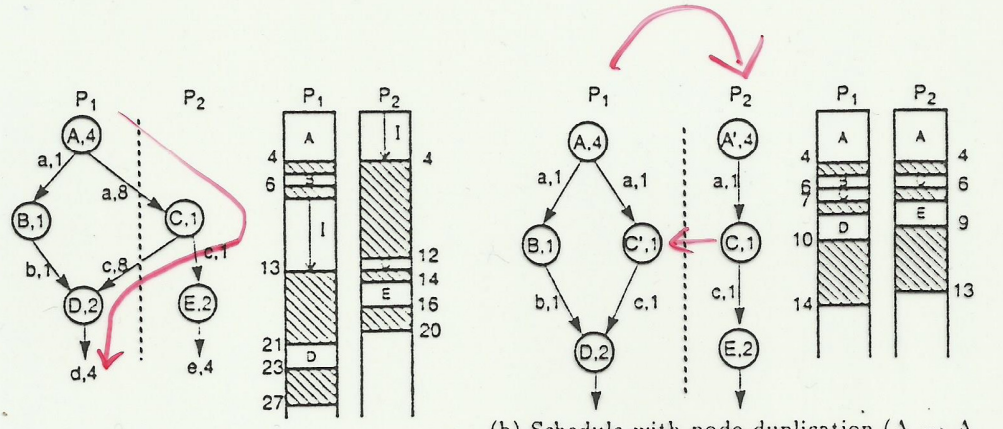
End



(a) Fine grain (Fig. 2.6a) (b) Coarse grain (Fig. 2.6b)

Figure 2.7 Scheduling of the fine-grain and coarse-grain programs. (I: idle time; shaded areas: communication delays)

NODE DUPLICATION

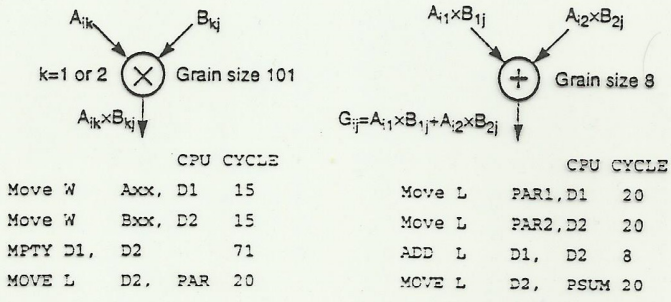


(a) Schedule without node duplication

(b) Schedule with node duplication (A → A and A′; C → C and C′)

Figure 2.8 Node-duplication scheduling to eliminate communication delays between processors. (I: idle time; shaded areas: communication delays)

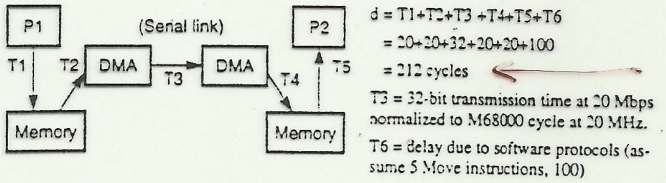
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$



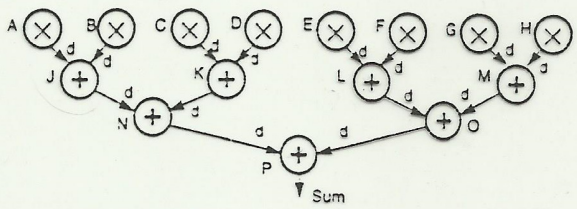

```
                    CPU CYCLE
Move W   Axx, D1    15
Move W   Bxx, D2    15
MPTY D1, D2         71
MOVE L   D2, PAR    20
```

```
                    CPU CYCLE
Move L   PAR1, D1   20
Move L   PAR2, D2   20
ADD  L   D1,   D2   8
MOVE L   D2,   PSUM 20
```

(a) Grain size calculation in M68000 assembly code at 20-MHz cycle



(b) Calculation of communication delay d



(c) Fine-grain program graph

Figure 2.9 Calculation of grain size and communication delay for the program graph in Example 2.5. (Courtesy of Kruatrachue and Lewis; reprinted with permission from *IEEE Software*, 1988)

Fig 2.10 "523" → "533"

2.11 (b).