$\boxed{\text{EE5902}}$ Solutions to Assignment Problems

(1.1)

$$CPI = \frac{45\times 1 + 32\times 2 + 15\times 2 + 8\times 2}{45+32+15+8} = 1.55 \ \text{cycles/instr}$$

$$\text{MIPS rate} = 10^{-6} \times \frac{40\times 10^{6}\ \text{cycles/sec}}{1.55\ \text{cycles/instru}}$$

$$= 25.8\ \text{MIPS}$$

$$\text{execution time} = \frac{\left(45000\times 1 + 32000\times 2 + 15000\times 2 + 8000\times 2\right)}{(40\times 10^{6})}$$

$$= \underline{\underline{3.875}}\ \text{msecs}$$

(or)

$$\text{execution time} = \frac{\text{Total \# of instructions}}{\text{MIPS rate}}$$

(1.3)

$$\text{Effective } CPI = \frac{15\times 10^{6}\ \text{cy/sec}}{10\times 10^{6}\ \text{instr/sec}} = 1.5\ \text{cycles/instr.}$$

Effective CPI of the new processor

$$= \underset{\text{one cycle delay}}{1} + \underbrace{0.3\times 2}_{\text{one mem access}} + \underbrace{0.05\times 4}_{\text{2 mem. accesses.}} = \underline{\underline{1.8\ \text{cy/instr}}}$$

PTO

(2) $\Rightarrow$ MIPS rate $= \frac{30\times10^6}{1.8} = \underline{16.7 \text{ MIPS}}$.

(1.4) Average CPI

(a) $= 1\times0.6 + 2\times0.18 + 4\times0.12$
$+ 8\times0.1 = 2.24$ cycle/instn.

(b) MIPS rate $= (40/2.24) = \underline{17.86 \text{ MIPS}}$

(1.5) (a) False

(b) True

(c) True

(d) False

(e) True.

(1.2) • Instruction set & compiler technology affect the length of the executable code & the mem. access frequency.

• CPU implementation & control determines the clock rate

• Mem. hierarchy impacts the effective mem.

access time.

- All these factors together determine the effective CPI, as explained in 1.1.4 (section).

1.8 (a) Total # of cycles needed on a sequential processor is

$$(4+4+8+4+2+4)\times 64$$

$$= 1664 \text{ cycles}$$

(b) Each PE executes the same instruction on the corresponding elements of the vectors involved. There is no communication among the processors. Hence, the total # of cycles on each PE is - $(4+4+8+4+2+4 = 26)$

(c) Speed-up = 64, with a perfectly parallel execution of the code.

[PTO]

$\boxed{\text{Bonus Prob 1}}$

Consider the computation of $\boxed{\bar{A}\bar{x} = B}$, where

$\bar{A}$ is a $m \times n$ matrix

$\bar{x}$: $n \times 1$ vector, yielding

$B = m \times 1$ vector.

You are given a network of p processors.

Communication from one processor to other takes

- α units of time for <u>one row</u> of the matrix, Let
- β denote the time taken for computation of one row of matrix (i.e; n multiplication and $(n-1)$ additions).

Possible strategies for this computation are_

- matrix $\bar{A}$ resides on each PE, but $\bar{x}$ can be split
- $\bar{x}$ can reside on each PE, but $\bar{A}$ can be split (row wise)
- $\bar{x}$ can reside on each PE, but $\bar{A}$ can be broadcasted to all PE
- $\bar{A}$ can reside on each PE, but $\bar{x}$ can be broadcasted.

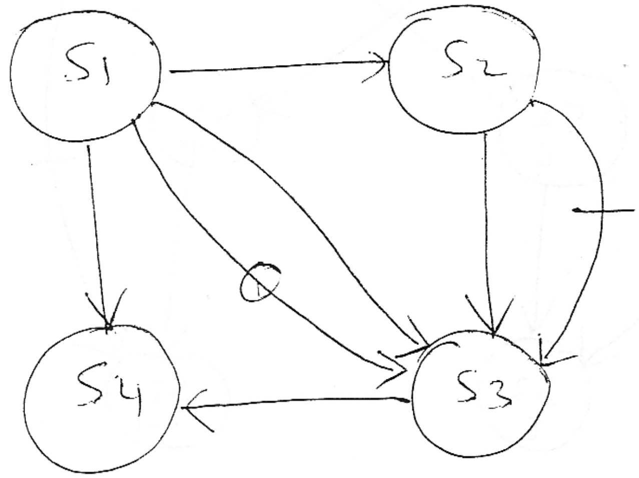
- You may also comprehend different strategies.

Q: Which strategy suits well under what conditions of communication & computation magnitudes?

Assume: (m/p) is an integer and row-wise partition for all your arguements.

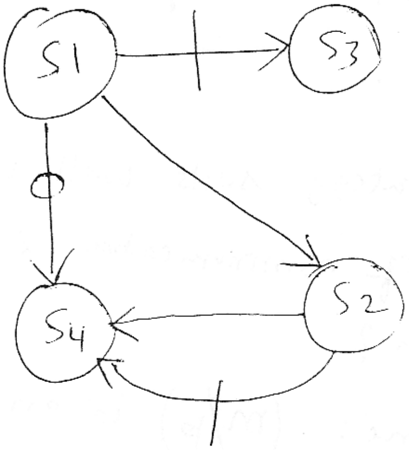
2.4

(a)

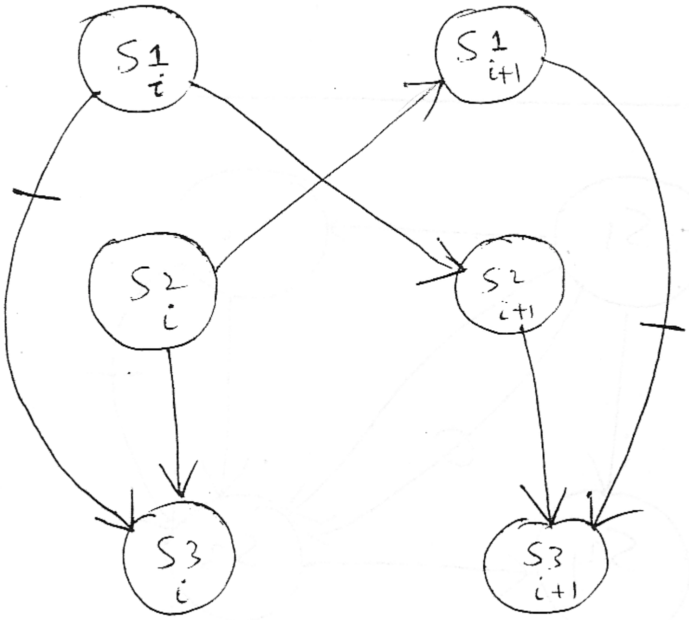


PTO

2.4 (b)



(c)

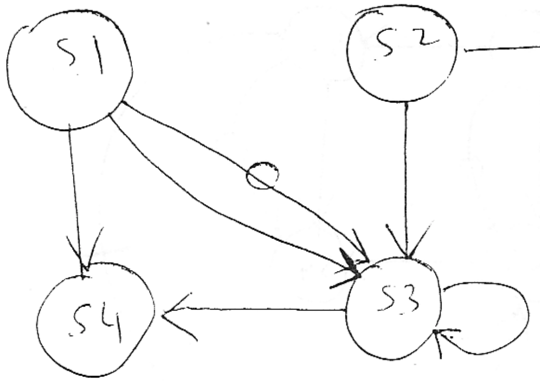


Note:

← Statement j , ith iteration.

2.5

(a)



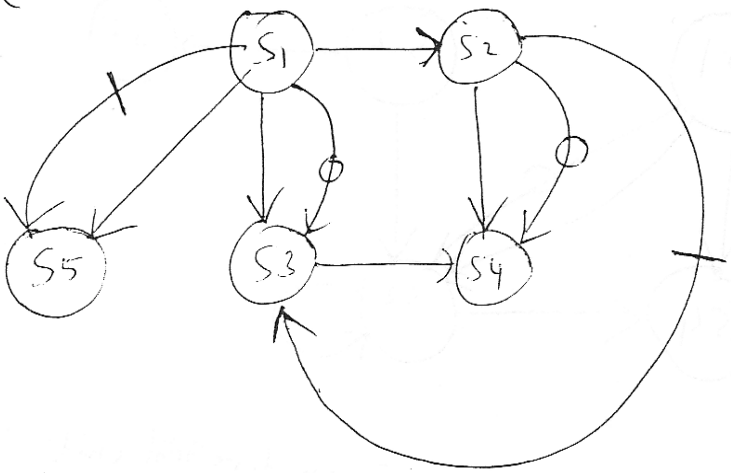
(b) If only one copy of each functional unit is available, there are storage dependences between instruction pairs (S2, S5) and (S4, S5).

There is also a resource dependence between S1 & S2 on the load unit and another between S4 & S5 on the store unit.

(c) (PTO)

(8)

(c)



There is an ALU dependence between S3 & S4 &

storage dependence between S1 & S5

(2.b) $I_1 = \{B, C\}$, $O_1 = \{A\}$

$I_2 = \{B, D\}$, $O_2 = \{C\}$

$I_3 = \phi$, $O_3 = \{S\}$

$I_4 = \{S, A, X(I)\}$, $O_4 = \{S\}$

$I_5 = \{S, C\}$, $O_5 = \{C\}$

Use Bernstein's conditions

- S_1 & S_3 can be executed concurrently, since $I_1 \cap O_3 = \phi$, $I_3 \cap O_1 = \phi$ & $O_1 \cap O_3 = \phi$

- S_2 & S_3 can be executed concur/.., since $I_2 \cap O_3 = \phi$, $I_3 \cap O_2 = \phi$ & $O_2 \cap O_3 = \phi$

- Similarly S_2 & S_4 can be executed concur/.

- - - - - - - - - -

- S_1 & S_5 cannot be executed concurrently, since $I_1 \cap O_5 = \{C\}$

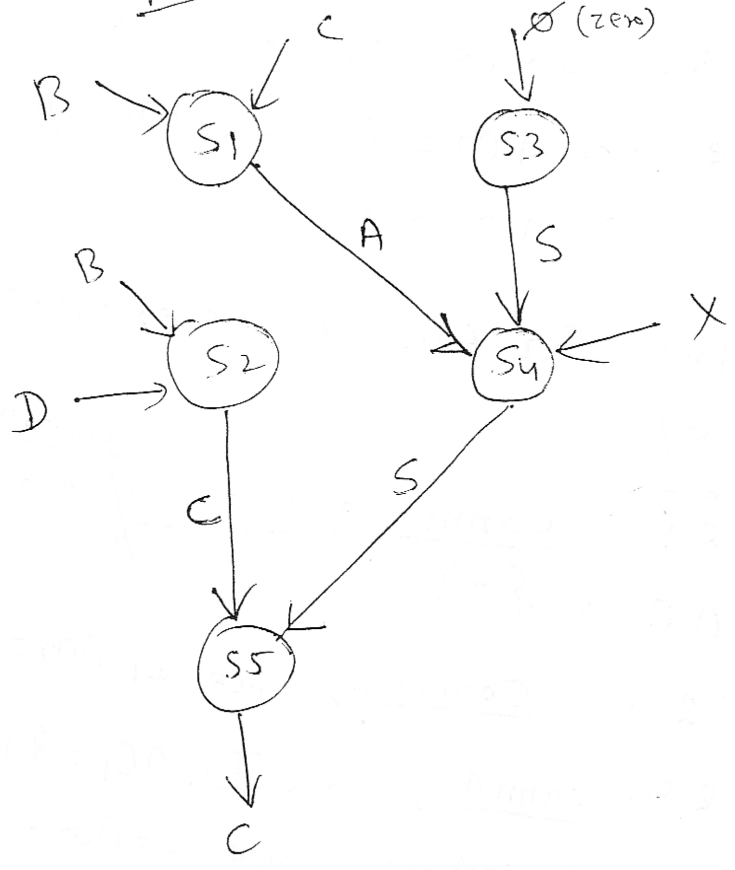
→ - S_1 & S_2 cannot ..., since $I_1 \cap O_2 = \{C$
- S_1 & S_4 cannot ..., since $I_4 \cap O_1 = \{A\}$
- S_2 & S_5 cannot ..., since $I_5 \cap O_2 = O_5 \cap O_2 = \{C$
- S_3 & S_4 cannot ..., since $I_4 \cap O_3 = O_4 \cap O_3 = \{S\}$
- S_3 & S_5 cannot ..., $I_5 \cap O_3 = \{S\}$

(PTO)

(10). S4 & S5 cannot ..., since

$$I_5 \cap I_4 = I_5 \cap D_4 = \{S\}$$

Restructured program is as follows



2.7

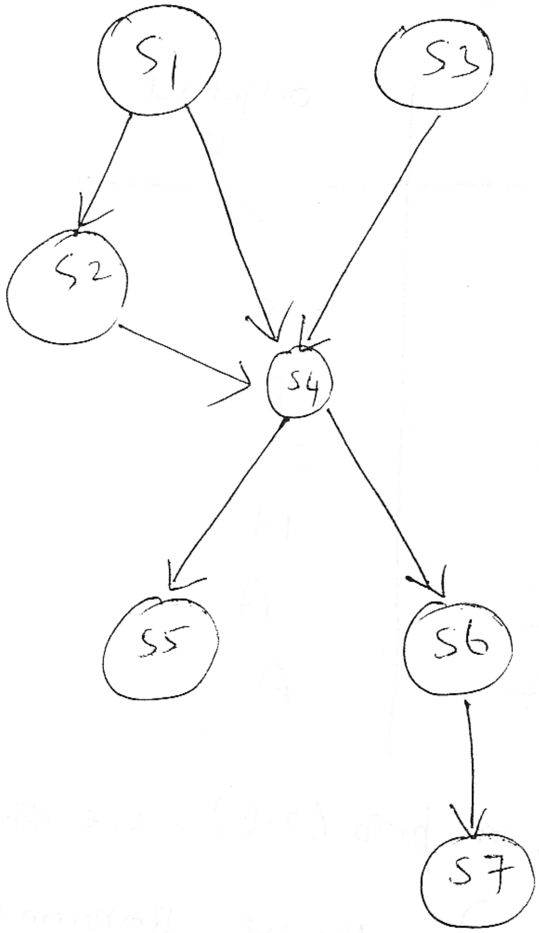
Instr.	input set I	output set O
1	B, C	A
2	D, E	C
3	G, E	F
4	A, F	C
5	G, C	M
6	L, C	A
7	E, A	A

Same procedure as in prob (2.6). we obtain,

S1 || S3,
S1 || S5,
S2 || S3,
S2 || S7,
S3 || S5,
S3 || S6
S3 || S7
S5 || S6
S5 || S7

~~However~~, Bernstein's conditions are not sufficient for this problem. We need to take care of the precedence relations. See the diagram below.

[P 10]



From this diagram, it is clear that statements S1 ,S2 & S3 must be executed before S4.

Thus, this consideration prohibits parallel execution among the two groups of statements.

Thus, we have only the following subset that can be executed parallely.

(P To)

S1 || S3

S2 || S3

S5 || S6 & S5 || S7.

Parallel code:

```
┌ Cobegin
│      S1, S3
└ Coend
  S2
  S4
┌ Cobegin
│      S5, S6
└ Coend
  S7
```

(4.4)

Type/item	CISC	RISC
• instruction format	16-64 bits per inst.	fixed (32 bit)
• addressing mode	12-24	limited to 3-5 (mostly register based, exept load/store)
• CPI	2-15, avg of 5	less than 1.5, close to 1

(b) & (c)

descriptive & can be found in the book.

4.11 average cost $c = \frac{c_1 s_1 + c_2 s_2}{s_1 + s_2}$

(a)

for $c \rightarrow c_2$, conditions are

$s_2 \gg s_1$ & $c_2 s_2 \gg c_1 s_1$

(b) effective access time,

$$t_a = \sum f_i t_i$$

$$= h_1 t_1 + (1-h_1) h_2 t_2$$

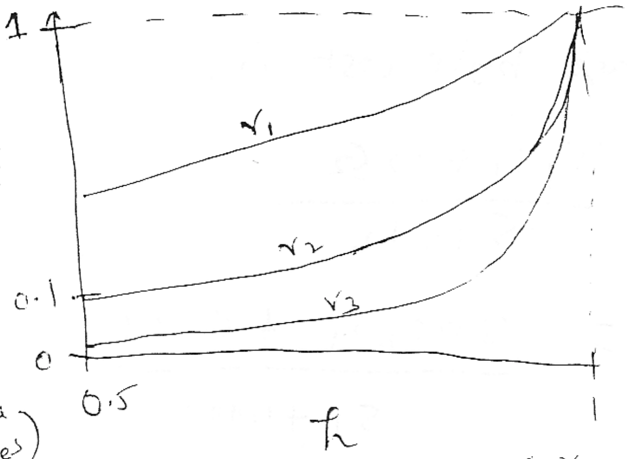
$$= h t_1 + (1-h) t_2$$

(c) If $t_2 = r.t_1$, then

$$t_a = \left[h + (1-h) r\right] t_1$$

$$E = (t_1/t_a) = \left[\frac{1}{h + (1-h) r}\right]$$

(d)



(please, plot with r = 5, 20 & 100 I am showing only the nature of the curves)

$r_1 < r_2 < r_3$.

(e) if $r = 100$,

$$E = \frac{1}{h + (1-h)100} > 0.95$$

$$\Rightarrow h > 99.95\%$$

4.12 $t_a = h_1 t_1 + (1-h_1) h_2 t_2$

$= h t_1 + (1-h) 10 t_1$

$= (10 - 9h) t_1$

if $h = 0.7$, $t_a = 3.7 t_1 = 3.7 \times 20$

$= \underline{74 \text{ ns}}$

if $h = 0.9$, $t_a = 1.9 t_1$

$= 38 \text{ ns}$

& 50 ns .

(PTO)

16 Average byte cost is,

$$= \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2}$$

$$= \frac{20 C_2 S_1 + C_2 \times 4}{S_1 + 4000}$$

$$\text{Avg. cost} = \frac{.4 S_1 + 800}{S_1 + 4000}$$

Use $S_1 = 64, 128,$ & 256 to get

Avg cost = 0.26, 0.32 & 0.43 respectively.

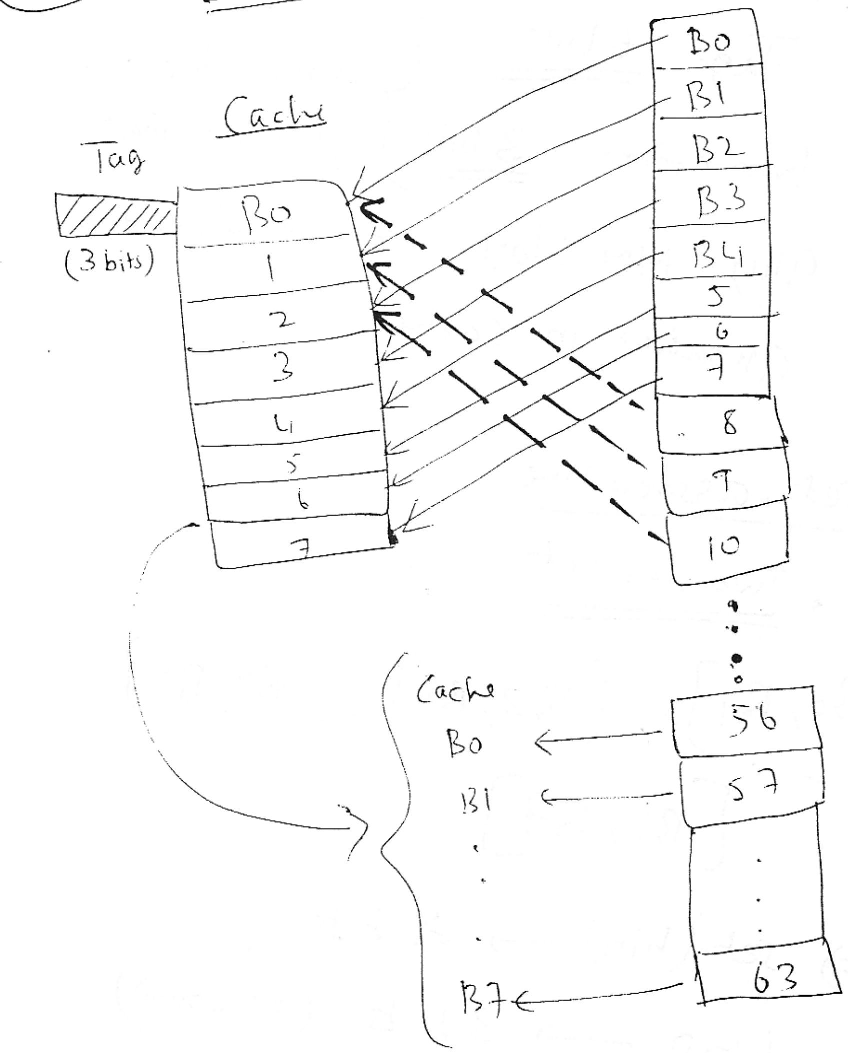
find the (average access time × averag cost)

You will get 19.24,
12.16,
10.15 , respectively.

Obviously we chose 10.15, the third option a, the best design choice.

(PTO)

5.8 Direct mapping:



3	3	3
Tag	block	word

remain (9-6) = 3 bits for Tag.

3 bits since cache has 8 blocks

block → 8 words

Total # of words in MM = 64x8 = $\underline{2^9}$

⇒ we need 9 bits to address a single location

(PTO)

(18) (b) Fully Associative mapping:

Tag 6 bits
Word address 3 bits } since the policy is that any MM block can be placed anywhere in cache.

(c) set-associative

- Two way set

[B0, B1], [B2, B3], [B4, B5]
[B6, B7]

⇒ set (bits) → 2 bits
Word → 3 bits (as usual)
⇒ 9 − 5 = 4 bits for Tag.

T	S	W
4	2	3

5.9

(a) Each set of the cache consists of

$$\left(\frac{256}{8}\right) = 32 \text{ blocks}$$ ✓

& the entire cache has,

$$16 \times \frac{1024}{256} = \underline{64 \text{ sets}}$$ ✓

Similarly, main memory contains, $\left[1024 \times 1024 / 8\right]$

$= \underline{131072}$ blocks

Thus,

2^{10} 2^{4} 2^{24}

Tag	Set	W
↓	↓	↓
15 bits	6 bits	3. bits ← 24 bits (total space)

Thus, a block K of the MM is mapped to a block ~~fa~~ in set F of the cache if,

$$\underline{F = K \bmod 64}$$

k = 1024

1024 mod 64

(P/o)

20

(b) Effective mem access time for this mem. hierarchy is,

$$50 \times 0.95 +$$
$$400 \times (1-0.95)$$
$$= 47.5 + 20 = \underline{67.5\ ns}$$

5.13

(a) $CPI = m \cdot t + \left(\frac{1}{x}\right)$

$$= \left(\frac{1 + mt\,x}{x}\right)$$

$$\Rightarrow MIPS = \left(\frac{p}{CPI}\right) = \left(\frac{px}{1+mtx}\right)$$

(b) with $p = 32$, $m = 0.4$, $t = 1\,\mu sec.$ when MIPS is 56 effective, what is MIPS rate of each processor?

$\rightarrow$ from (a) : $\dfrac{32 \cdot x}{1 + m \cdot t \cdot x}$

i.e.,

$$\text{MIPS} = \frac{32\,x}{1+0.4x} = 56$$

$$\Rightarrow \underline{\underline{x = 5.83}} \text{ in MIPS}$$

(c) Substitute in (a):

$$\frac{32 \times 2}{1+1.6 \times 1 \times 2} = \underline{15.24 \text{ MIPS}}$$

5.14 (a) $t_a = f_i(1-h_i)t_m + f_d(1-h_d)t_m$

$$CPI = m \cdot t_a + \frac{1}{x} + \alpha \cdot t_s$$

$$= \frac{x(m t_a + t_s) + 1}{x}$$

$$\text{effective MIPS} = \left(\frac{p}{CPI}\right) = \left(\frac{px}{x(m t_a + t_s) + 1}\right)$$

(b) $t_a = 0.0575$ (Substitute in (a))

& $CPI = 0.485$

(PTO)

(22) $\frac{p}{0.485} = 25$

$\Rightarrow \underline{p = 12}$

(c) cost of cache is

$4.7 \times 16(32+64) = \underline{7219.2}$

$\Rightarrow$ The total amt. of money allowed for the shared memory is 17781.8,

& mem. capacity in Mbytes is _

$C_m = \frac{17781.8}{0.4 \times 1024} \simeq \underline{43.4}$ (approx)

(6.1) (a) Speedup $= \frac{nk}{k+(n-1)} = 4.9986$

$\left[\because \frac{15000 \times 5}{5+(15000-1)}\right]$

Eff : $\frac{n}{k+(n-1)} = 0.99976$

Throughput = $\underline{24.99}$ MIPS (PTO)

6.4 $$PCR = \frac{1}{\left(\frac{t}{k}+d\right)(c+kh)}$$

Maximizing PCR is minimizing its inverse. Let $\underline{k^*}$ be the optimal # of pipeline stage.

$$\left[\frac{\partial}{\partial k}\left(\frac{1}{PCR}\right)\right]_{k^*} = 0$$

differential evaluated at k^*

$$\Rightarrow -\frac{t}{k^{*2}}(c+k^*h) + \left(\frac{t}{k^*}+d\right)h = 0$$

$$\Rightarrow \frac{c t}{k^{*2}} = dh$$

$$\Rightarrow k^* = \sqrt{\frac{c\cdot t}{d\cdot h}}$$

Note: $$\boxed{\left[\frac{\partial^2}{\partial k^2}\left(\frac{1}{PCR}\right)\right]_{k^*} > 0}$$

2nd differential evaluated at k^* must be >0 for minimization

This is important to realise & You should verify it!

PTO

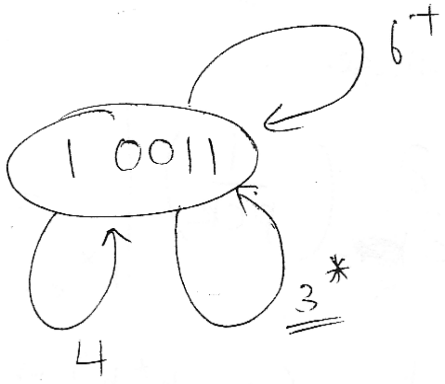
24

6b

(a) FL: { 1, 2, 5 }

Coll. vector (initial) = 10011

(b)



⇒

(c) MAL = 3

(d) Throughput = $\frac{1}{3\cdot 2}$ = 16.67 MOPS

Million Operations per sec.

(e) Lower bound on MAL = 2

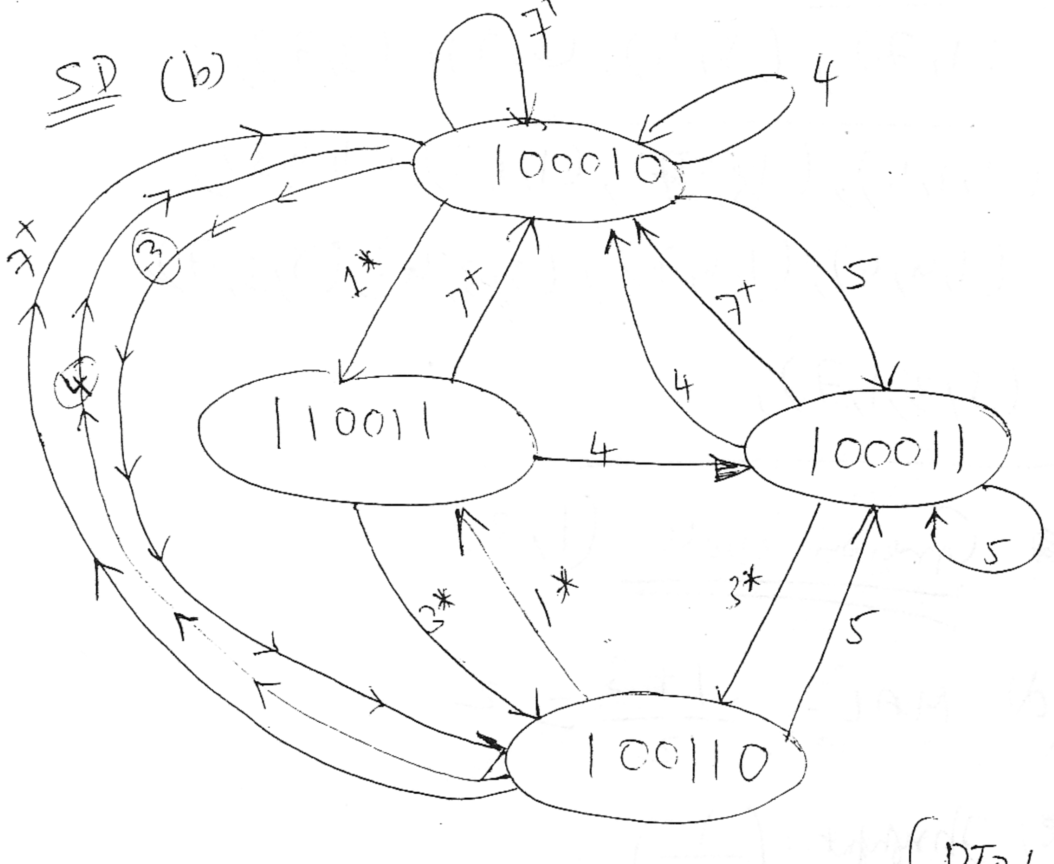
⇒ Optimal latency is not achieved

(PTO)

6.7

(a)	1	2	3	4	5	6	7
S_1	X						X
S_2		X					
S_3			X				
S_4				X		X	
D					X		

SD (b)



(PTO)

(25) Simple cycles: (Oh God!!)

(c) (4),

(5),

(7),

(3, 1),

(3, 4)

(3, 5, 4)

(3, 5, 7)

(1, 7), (5, 4), (5, 7), (3, 7),

(1, 3, 4), (1, 3, 5, 7), (1, 3, 7), (1, 4, 3),

(1, 4, 4), (1, 4, 7), (5, 3, 4), (5, 3, 7)

(5, 3, 1, 7),

Greedy cycle: (1, 3)

(d) MAL - $\frac{1+3}{2} = 2$

(e) Throughput : $\left(\frac{1}{2\tau}\right)$

6.9 → Same standard procedure, please!

6.13 Note that we are constrained to use only 6 columns & at the 6th column

	1	2	3	4	5	6
S_1	X				X	
S_2		X				X
S_3			X			
S_4				X		

we need the output from S2. Thus we have a 'X' mark in (S2, 6) cell. Going backwards, we can see the RT as shown above:

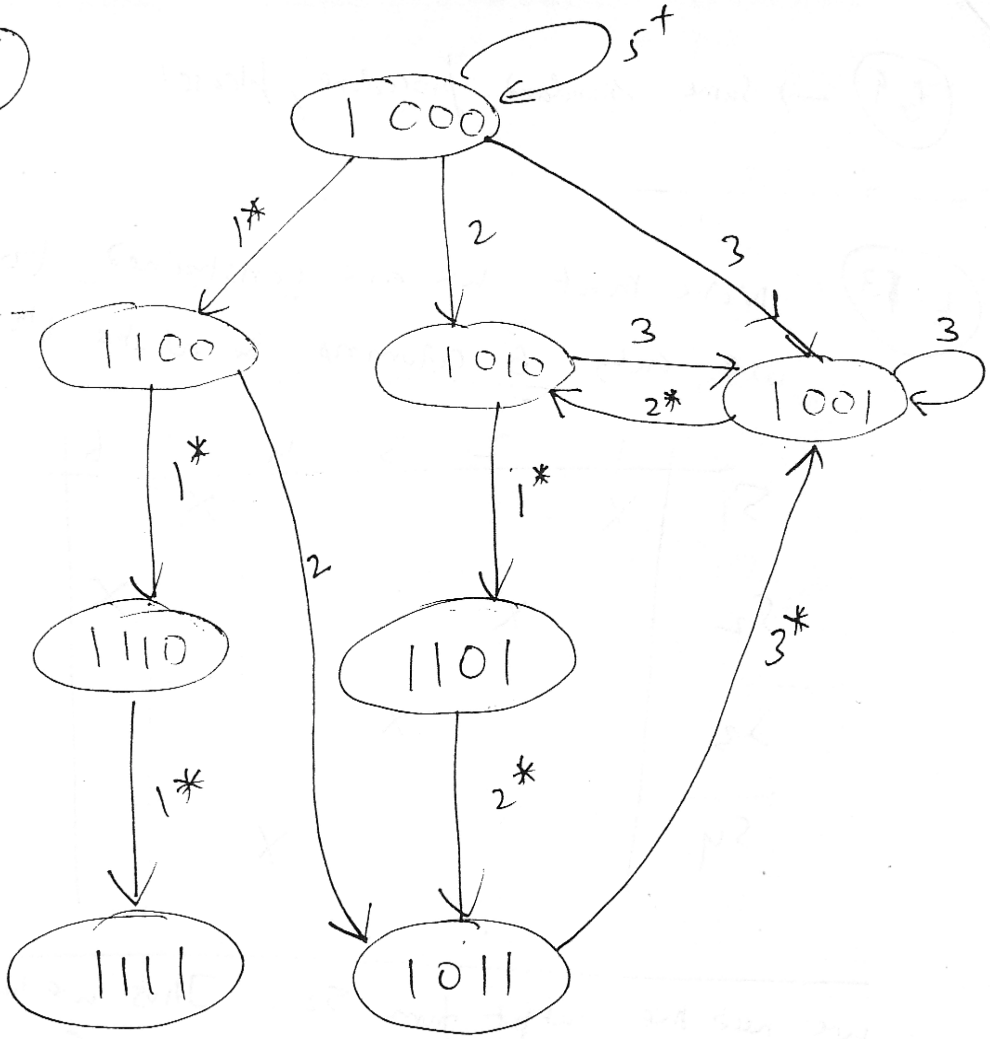
Then, (b), (c), (d), (e), (f), (g) follow — standard procedure!

(e) (1,1,1,5) & (1,2,3,2)

(f): MAL = 2. &

(for S.Diagram — PTO)

(PTO)

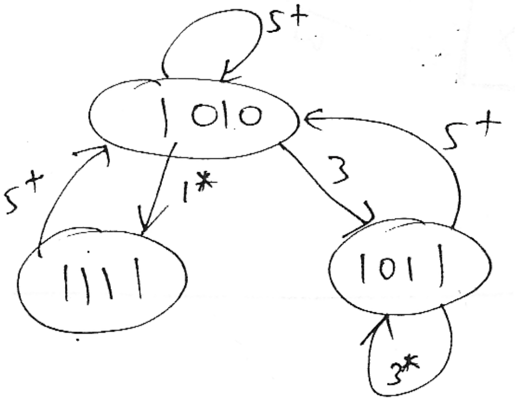


From each state to 1000 put a link with weight 5+ (I am avoiding this so that the picture is clear!)

6.13 Solution 2 : (other possibility).

	1	2	3	4	5	6
S1	X					
S2		X		X		X
S3			X		X	
S4				X		

S_D:



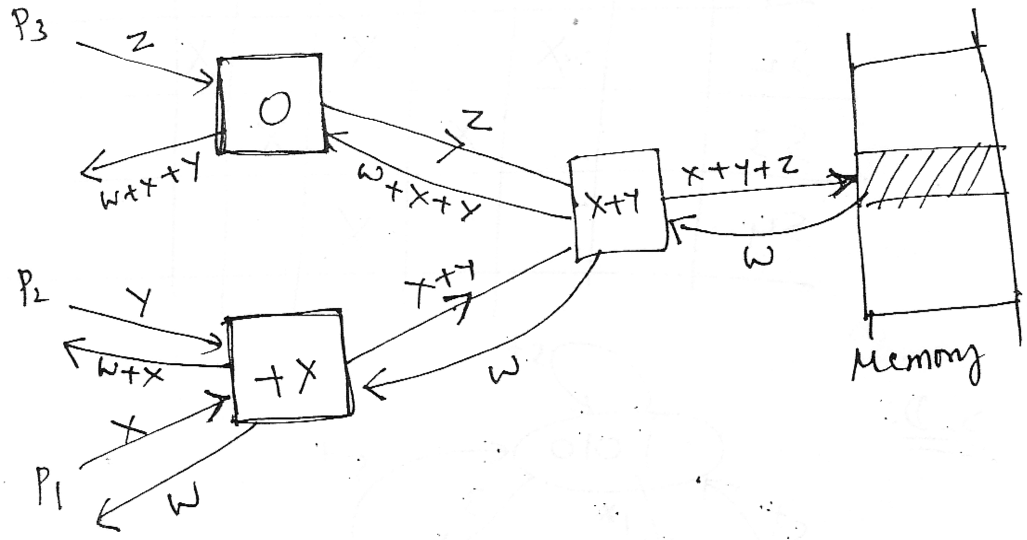
Simple cycl : (3), (5), (1,5), (3,5)

MAL = 3 as greedy cycles {(1,5), (3)}

max. throughput = $\left(\frac{1}{3\tau}\right)$

Good luck from Bharadwaj, V

Solution for the fetch-and-add problem :


P3
Z
0
Z
W+X+Y
W+X+Y
X+Y
X+Y+Z
W
Memory
P2
Y
W+X
X+Y
+X
W
X
P1
W