Shahzeb Qamar
18BCS032

Parallel & Distributed Computing
Sessional - 1

① S1 : A = B + C
S2 : C = B × D
S3 : S = 0
S4 : Do I = A, 100
S = S + X(i)
End Do
S5 : IF (S. GT. 1000) C = C × 2

Bernstein's condition of parallelism ⇒
$I_1 \cap O_2 \neq \phi$ (Anti Independence)
$I_2 \cap O_1 \neq \phi$ (Flow Independence)
$O_1 \cap O_2 \neq \phi$ (Output Independence)

	I	O
S_1	B, C	A
S_2	B, D	C
S_3	ϕ	S
S_4	A, S, I, X	S
S_5	S, C	C

For S_1 and S_2
$I_1 \cap O_2 = C$, $I_2 \cap O_1 = \phi$, $O_1 \cap O_2 = \phi$
Anti-dependence, so not executed in parallel

For S_1 and S_3
$I_1 \cap O_3 = \phi$, $I_3 \cap O_1 = \phi$, $O_1 \cap O_3 = \phi$
$S_1 \parallel S_3$

For S_1 and S_4
$I_1 \cap O_4 = \phi$, $I_4 \cap O_1 = A$, $O_1 \cap O_2 = \phi$
can't be executed in parallel, Flow independence

Shahzeb Qamar
18BCS032

For S_1 and S_5

$I_1 \cap O_5 = C$ $I_5 \cap O_1 = \phi$ $O_1 \cap O_5 = \phi$

can't be executed in parallel

For S_2 and S_3

$I_2 \cap O_3 = \phi$ $I_3 \cap O_2 = \phi$ $O_2 \cap O_3 = \phi$

$S_2 \parallel S_3$

For S_2 and S_4

$I_2 \cap O_4 = \phi$ $I_4 \cap O_2 = \phi$ $O_2 \cap O_4 = \phi$

$S_2 \parallel S_4$

For S_2 and S_5

$I_2 \cap O_5 = \phi$, $I_5 \cap O_2 = C$, $O_2 \cap O_5 = \phi$

can't be executed in parallel

For S_3 and S_4

$I_3 \cap O_4 = \phi$ $O_3 \cap I_4 = S$ $O_3 \cap O_4 = S$

can't be executed in parallel

For S_3 and S_5

$I_3 \cap O_5 = \phi$ $I_5 \cap O_3 = S$, $O_3 \cap O_5 = \phi$

can't be executed in parallel

For S_4 and S_5

$I_4 \cap O_5 = \phi$, $I_5 \cap O_4 = S$, $O_4 \cap O_5 = \phi$

can't be executed in parallel

Pairs than can be executed in parallel =>

$S_1 \parallel S_2$; $S_2 \parallel S_3$; $S_2 \parallel S_4$

P_1 = A = B + C
C = B * D

P_2 = S = 0
DO I = A, 100
S = S + X(I)
END DO
C = C * 2

4K (5.5T. 1000)

Shanzeb Kumar
18BCS032

Restructured ⇒

P1 = A = B + C S = 0
P2 ⇒ C = B * D Seed DO I = A, 100
S = S + X(I)
END DO
P3 ⇒ IF (S.GT. 1000) C = C * 2

Shahzeb
18B CS032

As per Amdahl's law,

(2) (a) $Speedup = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$

$\Rightarrow$

$Speedup_{enhanced} = 10$

$Fraction_{enhanced} = 0.5$

$$Speedup = \frac{1}{(1-0.5) + \frac{0.5}{10}}$$

$$= \frac{1}{0.5 + 0.05}$$

$$= \frac{1}{0.55}$$

$$= 1.818$$

(a) i) Without enhancement, time taken as a = 0.5 (50%)
Accelerate/enhanced phase would take = 5 (500%)

Relative enhancement = 5 + 0.5 = 5.5 (550%)

$$Overall\ speedup = \frac{execution_{unaccelerated}}{execution_{accelerated}} = \frac{5.5\ (550\%)}{1\ (100\%)}$$

$$= 5.5$$

(b) Percentage of original execution time which has accelerated =

$$Fraction_{revised} = \frac{speedup_{overall} \times speedup_{accelerated} - speedup_{accelerated}}{speedup_{overall} \times speedup_{accelerated} - speedup_{overall}}$$

$$= \frac{5.5 \times 10 - 10}{5.5 \times 10 - 5.5}$$

$$= 90.90\%$$

Shahzeb Qamar
18BCS032

(3) Forbidden Latencies ⇒

f1 ⇒
row1 = 3-1 = 2

f2 ⇒
row3 = 4-3 = 1

f3 ⇒
row1 = 4-1 = 3

Forbidden latencies = 1, 2, 3

Collision vector = 111

State transition diagram ⇒

Simple cycle = 4
Greedy cycle = 4
MAL = 4

Shahzeb Qamar
18BCS032

(6) Collision vector = 1, 2, 3, 7, 8, 9

(7) In Tomasulo's algorithm 2 different techniques are combined the renaming of two the architectural registers to a larger set of registers and buffering of source operands from register file.

eg ⇒ DIV F_0, F_2, F_4
MUL F_8, F_0, F_8
ADD F_{12}, F_8, F_{14}

MUL doesn't have a dependency and can be executed before MUL instead of waiting for stall. MUL needs to be stored in a buffer called reservation station.

To allow MUL to proceed when its operand becomes available, the RS must be informed when result is available.

Results are broadcasted on common data bus to all RS

Pipeline phases:

IF ⇒ fetch next instruction into FIFO queue.

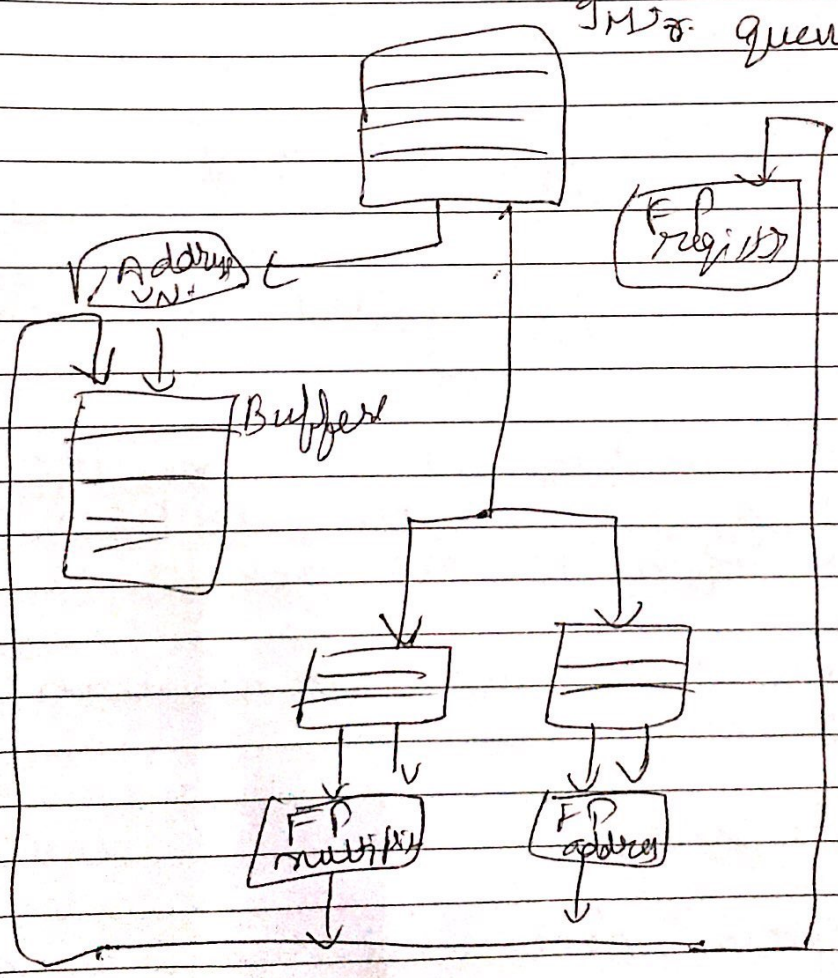
Issue: Get instruction from head of queue if matching RS is free of (there is no structural hazard), issue instruction

Shahzeb Qamar
18BCS032

to R.S. write operand values if available otherwise with identifiers

Execute : when all operands are available (no RAW dependence) and functional unit is available, then monitor data bus for result.

④ Write result ⇒ writes result on data bus, to all waiting stations and reg. file



Shahzeb Samar
18BCS032

(5) Discuss (m,n) branch predictor

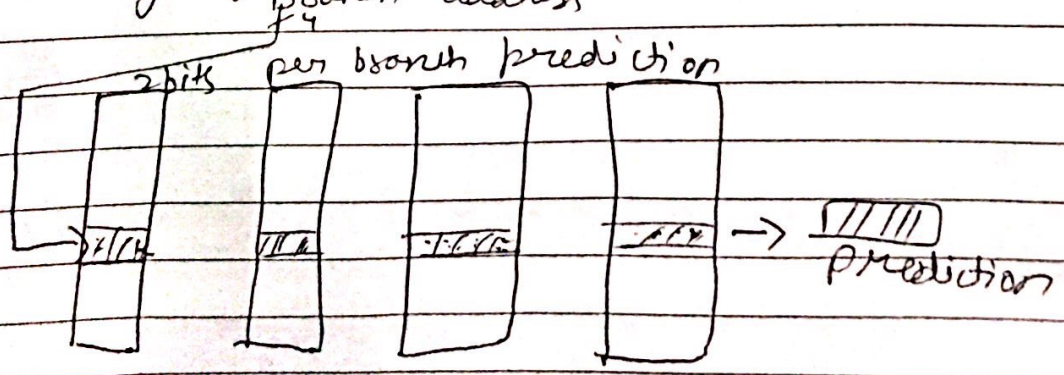
The (m,n) branch predictor is also known as correlated branch prediction. It is a two level branch prediction in which prediction accuracy is improved as it takes into consideration the recent behaviour of other branches also.

1. It uses k least significant bits of branch target address which is fetched before

2. It also uses local history table (LHT) which is table of shift registers where shift registers refers to the last outcome of m branches having same k least significant bits.

3 It also uses local prediction table to predict the outcome depending on the state in which it is present

(2,2) predictor
→ behaviour of recent branches selects between four prediction of next branch updating just the prediction



For example, (2,2) branch prediction
$2^2 \times 2 \times$ No. of entries = 8K bits
No. of entries = 8K/8 = 1K