Chapter 7. Concurrent Processors

Problem 7.1

VADD VR3, VR1, VR2

VMPY VR5, VR3, VR4

Vector size = 64

$$\begin{aligned} \text{Unchained time} &= 8 \text{ (add startup)} + 64 \text{ (elements/VR)} + 8 \text{ (mpy startup)} + 64 \text{ (elements)} \\ &= 144 \text{ cycles.} \\ \text{Chained time} &= 8 \text{ (add startup)} + 8 \text{ (mpy startup)} + 64 \text{ (elements/VR)} = 80 \text{ cycles.} \end{aligned}$$

a. Implied (average) instruction memory bandwidth

$$= \frac{2 \text{ inst} \times 4 \text{ bytes/inst}}{144 \text{ cycles}} = .0556B/\text{cycle for unchained}$$

and

$$= \frac{2 \text{ inst} \times 4 \text{ bytes/inst}}{80 \text{ cycles}} = .1B/\text{cycle for chained}$$

b.

```
VLD   VR1,  add1
VLD   VR2,  add2
VLD   VR4,  add3
VADD  VR3,  VR1,  VR2
VMPY  VR5,  VR3,  VR4
VST   VR5,  add4
```

$$\begin{aligned} \text{Unchained time} &= (8+8+8) \text{ (load startup)} + 64 \text{ (load completion)} + 8 \text{ (add startup)} \\ &\quad +64 \text{ (add completion)} + 8 \text{ (multiply startup)} + 64 \text{ (multiply completion)} \\ &\quad +8 \text{ (store startup)} + 64 \text{ (store completion)} \\ &= 6 \times 8 + 4 \times 64 = 304 \text{ cycles} \end{aligned}$$

A total of 64 registers $\times$ 4 instructions $\times\ 8^B = 2048$ bytes are moved.

Implied (average) data bandwidth $= \frac{2048 \text{ bytes}}{304 \text{ cycles}} = 6.7\text{B/cycles}$

$$\begin{aligned} \text{Chained time} &= (8+8+8) \text{ (load startup)} + 64 \text{ (load completion)} + 8 \text{ (add startup)} \\ &\quad +8 \text{ (multiply startup)} + 64 \text{ (multiply completion)} + 8 \text{ (store startup)} \end{aligned}$$

+64 (store completion)

$$= 6 \times 8 + 3 \times 64 = 240 \text{ cycles}$$

Implied (average) data bandwidth $= \frac{2048 \text{ bytes}}{240 \text{ cycles}} = 8.5\text{B/cycles}$

Problem 7.2

a. $F37B90_{16} = 330313232100_4$

$r = (0 - 0 + 1 - 2 + 3 - 2 + 3 - 1 + 3 - 0 + 3 - 3) \bmod 5 = 5 \bmod 5 = 0$

So, module address = 0

$$\begin{array}{rl} & 330313232100_4 \\ - & 000230211000_4 \\ \hline & 030023021100_4 = 30B250_{16} \end{array}$$

So, address in module = $30B250_{16}$

b. $AA3347_{16} = 222203031013_4$

$r = (3 - 1 + 0 - 1 + 3 - 0 + 3 - 0 + 2 - 2 + 2 - 2) \bmod 5 = 7 \bmod 5 = 2$

So, module address = 0

$$\begin{array}{rl} & 222203031013_4 \\ - & 202002210012_4 \\ \hline & 020200221001_4 = 220A41_{16} \end{array}$$

So, address in module = $220A41_{16}$

Problem 7.3

a.

m_3	m_2	m_1	m_0	m'_3	m'_2	m'_1	m'_0
0	0	0	0	0	0	1	0
0	0	0	1	0	1	1	1
0	0	1	0	1	0	0	0
0	0	1	1	1	1	0	1
0	1	0	0	0	1	1	0
0	1	0	1	0	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	1	1	1	1
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	1	1	1	0
1	1	0	1	1	0	1	1
1	1	1	0	0	1	0	0
1	1	1	1	0	0	0	1

Since each original address maps to a unique hashed address, the scheme works.

b. $F37B90_{16}$

The 4 least significant bits are 0000_2 which map to 0010_2, thus the module address is 2.

$AA3347_{16}$

The 4 least significant bits are 0111_2 which map to 1001_2, thus the module address is 9.

Problem 7.4

$\Delta T = 8\,\text{ns}$

Memory cycle time (T_c) = 64 ns

Two requests per ΔT

a. Offered memory bandwidth

$n = 2 \times \frac{64}{8} = 16$

$\lambda = \frac{n}{T_c} = \frac{16}{64 \times 10^{-9}} = 250$ MAPS

Assume 64 bit word size

$\lambda = 250$ MAPS $\times$ 8 bytes $= 1907$ MBps

b. Achieved bandwidth

Use $M_B/D/1$

$$\begin{aligned} B(m,n) &= 8 + 16 - \frac{1}{2} - \sqrt{(8+16-.5)^2 - 2 \times 8 \times 16} = 6.288 \\ \lambda_a &= B/n \times \lambda_{\text{offered}} \\ &= \frac{6.288}{16} \times 1907 \text{ MBps} \\ &= 750 \text{ MBps} \end{aligned}$$

c. Mean queue size

$Q_{c-t} = n - B = 16 - 6.288 = 9.712$

Problem 7.8

$\gamma = .5$, $m = 17$

$n = 16$ and $T_c = 64$ ns, from problem 7.4

Assume 64 bit word size.

$B(m,n,\gamma) = 17 + 16(1+.5) - .5 - \sqrt{(17+16(1+.5)-.5)^2 - 2 \times 16 \times 17(1.5)} = 11.79$

$\lambda_{\text{ach}} = \frac{B}{T_c} = \frac{11.79}{64 \times 10^{-9}} \approx 184$ MAPS $= 1405$ MBps

Problem 7.10

4 memory requests/ΔT

$\Delta T = 8$ ns

$T_c = 42$ ns

a. Minimum number of interleaving for no contention

$m > n$

$m > 4 \times \frac{42}{8} = 21$, or $m = 32$ if only powers of 2 are allowed.

b. $m = 64$

$$\begin{aligned} \gamma_{\text{opt}} &= \frac{n-1}{2m-2n} = \frac{21-1}{2 \times 64 - 2 \times 21} = .233 \\ \text{Mean TBF} &= n \times \gamma_{\text{opt}} = 21 \times .233 = 4.893 \\ B(m,n,\gamma) &= B(64, 21, .233) = 20.567 \\ \text{Relative perf} &= \frac{20.567}{21} = .979 \end{aligned}$$

Problem 7.11

Assume that both the vector processor and the uniprocessor have the same cycle time. Assume that the uniprocessor can execute one instruction every cycle. Assume that the vector processor can do chaining.

The vector processor can load 3 operands concurrently in advance of using them since there are 3 read ports to memory. It can also store a result concurrently since there is a write port to memory. With chaining, the vector processor can perform 2 arithmetic operations concurrently if there are enough functional units.

Thus, the vector processor can perform 6 operations per cycle and the maximum speedup over a uniprocessor is 6.

Problem 7.14

```
I1   DIV.F   R1,R2,R3
I2   MPY.F   R1,R4,R5
I3   ADD.F   R4,R5,R6
I4   ADD.F   R5,R4,R7
I5   ST.F    ALPHA,R5
```

a. Improved control flow.

Assume there is no reorder buffer.

Assume floating point divide, multiply, and add take 8,4, and 3 separately.

Cycle 1: Decoder Issues I_1 →DIV unit
R2,R3 →DIV Res Stn
TAG_DIV →R1
Cycle 2: Divide begins DIV.F
Cycle 9: Divide completes
Divide requests permission to broadcast result in the next cycle
Cycle 10: Divide Result →R1
Cycle 11: Decoder Issues I_2 →MPY unit
R4,R5 →MPY unit
TAG_MPY →R1
Cycle 12: MPY begins
Decoder issues I_3 →ADD unit
R5,R6 →ADD unit
ADD_TAG1 →R4
Cycle 13: ADD begins
Decoder issues I_4 →ADD unit
ADD_TAG1 →ADD unit
R7 →ADD unit
ADD_TAG2 →R5
Cycle 14: ADD2 waits
Decoder issues I_5 →store unit
ADD_TAG2 →store units
Cycle 15: I_2, I_3 completes, requests for broadcast
(I_2 is granted, I_3 is not granted)
Cycle 16: MPY →R1
Cycle 17: ADD →R4, ADDER2's Res Stn

Cycle 18: ADD (I_4) begins
Cycle 20: I_4 completes and requests for broadcast (granted)
Cycle 21: ADD →R5, store units
Cycle 22: I_5 begins

b. Data flow with value holding reservation station.

Assume reorder buffer.

Cycle 1:	Decoder issues	I_1 →DIV unit
		R3,R2 →DIV Res Stn
		TAG_DIV →R1
Cycle 2:	Divides begins	DIV.F
	Decoder issues	I_2 →MPY unit
		R4,R5 →MPY Res Stn
		TAG_MPY →R1
Cycle 3:	Begin	MPY.F
	Decoder issues	I_3 →ADD unit
		R5,R6 →ADD Res Stn
		TAG_ADD1 →R4
Cycle 4:	Begin	ADD.F
	Decoder issues	I_4 →ADD unit
		TAG_ADD1 →ADD Res Stn
		R7 →ADD Res Stn
		TAG_ADD2 →R5
Cycle 5:	ADDER for I_4 waits	
	Decoder issues	I_5 →Store unit
	TAG_ADD2 →store buffer	
Cycle 6:	MPY completes and requests for broadcast (granted)	
	ADD completes and requests for broadcast (not granted)	
Cycle 7:	Multiply unit →R1	
Cycle 8:	ADD unit (1st unit) →R4,2nd Res Stn of Adder	
Cycle 9:	ADD (I_4) begins	
	DIV completes and requests for broadcast (granted)	
Cycle 10:	DIV.F broadcasts but ignored	
Cycle 11:	ADD completes and request for broadcast (granted)	
Cycle 12:	ADD unit →R5, store buffer	
Cycle 13:	STORE begins	

c. Control flow with shadow register.

Assume reorder buffer.

In this case, it would be better to rewrite the code using register renaming.

DIV.F	R1,R2,R3	renamed
MPY.F	R9,R4,R5	R1→R9
ADD.F	R10,R5,R6	R4→R10
ADD.F	R11,R10,R7	R5→R11
ST.F	ALPHA,R11	R5→R11

Cycle 1: Decoder issues I_1 →DIV unit
R3,R2 →DIV Res Stn
TAG_DIV →R1
Cycle 2: Divides begins DIV.F
Decoder issues I_2 →MPY unit
R4,R5 →MPY Res Stn
TAG_MPY →R9
Cycle 3: Begin MPY.F
Decoder issues I_3 →ADD unit
R5,R6 →ADD Res Stn
TAG_ADD1 →R10
Cycle 4: Begin ADD.F
Decoder issues I_4 →ADD unit
TAG_ADD1 →ADD Res Stn
R7 →ADD Res Stn
TAG_ADD2 →R11
Cycle 5: ADDER for I_4 waits
Decoder issues I_5 → Store unit
TAG_ADD2 → store buffer
Cycle 6: MPY completes and requests for broadcast (granted)
ADD completes and requests for broadcast (not granted)
Cycle 7: Multiply unit → R9
Cycle 8: ADD unit (1st unit) → R10, 2nd Res Stn of Adder
Cycle 9: ADD (I_4) begins
DIV completes and requests for broadcast (granted)
Cycle 10: DIV.F → R1
Cycle 11: ADD completes and request for broadcast (granted)
Cycle 12: ADD unit → R11, store buffer
Cycle 13: STORE begins