

## CPE 631 Advanced Computer Systems Architecture: Homework #4

### Practical Work

**Question #1 (60 points)** Evaluate effectiveness of the blocking optimization on a simulated model of the processor using SimpleScalar toolsuite (sim-cache and sim-outorder simulators).

A. Write a C subroutine for matrix multiplication  $MC = MA \times MB$ :

```
void mm(double **ma, double **mb, double **mc, int d);
```

Matrices ma, mb, and mc are squared with  $d \times d$  elements of type *long long int*. Let d be an input parameter for your main program that initializes the matrices and calls the subroutine for matrix multiplication.

B. Modify your program from the part A to support the blocking optimization technique. Let b (blocking factor) be an additional input parameter of your program.

C. Compare performance of the base program from A and the program from B for different blocking factors b assuming  $n=128$  (or 256)?

Note: The CPU model is as follows: L1I (8 KB, 4-way set-associative, 32 B cache line) + L1D (8 KB, 4-way set associative, 32 B cache line), L2U (64 KB, 32 B cache line, LRU replacement policy, 4-way set-associative). Assume memory latency of 240 clock cycles for the first chunk, and 4 processor clock cycles for the following chunks.

### Theory

#### **Question #2: (20 points)**

Consider the following code sequence executing on a CPU using the Tomasulo's algorithm for a single-issue processor. The loop implements  $Y = aX/Y$  for a vector of length 100. Initially  $R1=0$  and  $F0$  contains a.

```
foo:  L.D          F2,0(R1)          ; load X(i)
      MUL.D     F4,F2,F0           ; multiply a*X(i)
      L.D          F6,800(R1)       ; load Y(i)
      DIV.D     F6,F4,F6           ; a*X(i)/Y(i)
      S.D          F6,800(R1)       ; store Y(i)
      DADDUI    R1,R1,#8           ; increment R1
      DSGTUI    R3,R1,800          ; test if done
      BEQZ     R3,foo              ; loop if not done
```

Assume the following:

The EX stage does the effective address calculation for loads and stores.

Loads take 1 clock cycle.

Results are communicated via a single CDB.

The issue (IS) and write back (WB) stages take 1 clock cycle.

There are separate integer units for effective address calculation, for ALU operations, and for branch condition evaluation.

Functional units are not pipelined. There are 5 load buffer slots. There are 5 store buffer slots

FU Type	Cycles in EX	Number of FUs	Number of reservation stations
Integer (eff. address)	1	1	2
Integer (ALU oper.)	1	1	2
Integer (branch eval.)	1	1	2
FP adder	4	1	3
FP multiplier	8	1	3
FP divider	15	1	2

Fill the following table assuming Tomasulo's algorithm with speculation for the first 3 iterations of the loop.

Iter. number	Instructions	ISSUE (at clock)	EX (start-stop)	Read Access (at clock)	Write on CDB (at clock)	Commits (at clock)	Comment
1	L.D F2,0(R1)						
1	MUL.D F4,F2,F0						
1	L.D F6,800(R1)						
1	DIV.D F6,F4,F6						
1	S.D F6,800(R1)						
1	DADDUI R1,R1,#8						
1	DSGTUI R3,R1,800						
1	BEQZ R3,foo						
2	L.D F2,0(R1)						
2	MUL.D F4,F2,F0						
2	L.D F6,800(R1)						
2	DIV.D F6,F4,F6						
2	S.D F6,800(R1)						
2	DADDUI R1,R1,#8						
2	DSGTUI R3,R1,800						
2	BEQZ R3,foo						
3	L.D F2,0(R1)						
3	MUL.D F4,F2,F0						
3	L.D F6,800(R1)						
3	DIV.D F6,F4,F6						
3	S.D F6,800(R1)						
3	DADDUI R1,R1,#8						
3	DSGTUI R3,R1,800						
3	BEQZ R3,foo						

**Question #3: (20 points)** Consider the following code sequence executing on a double-issue CPU using the Tomasulo's algorithm with speculation.

```

Loop: LD.D  F4, 0(R1)           ; load X(i)
      DIV.D F0, F4, F2         ; X(i)/a
      LD.D   F8, 0x1000(R1)    ; load Y(i)
      ADD.D F6, F0, F8         ; X(i)/a + Y(i)
      S.D   F6, 0(R1)         ; store new X(i)
      DADDIU R1, R1, #4        ; increment pointer
      BNE   R1, R3, Loop      ; end of loop

```

Assume that execution of the DIV.D instruction requires 8 clock cycles while ADD.D requires 3 clock cycles. Assume that there is one integer unit (for integer execution and address calculation) and one unit for branch condition evaluation.

Fill the following table entering the clock cycle when the instructions issue, execute (from-to), read access, write the result back into the register file, and commit. Assume a separate pipelined functional unit for each FP operation. Assume that only one instruction may commit and that only one CDB is available.

Iter.	Instruction	Issue	Execute	Read access	WriteBack	Commit	Comment
1	LD.D	1					
1	DIV.D	1					
1	LD.D						
1	ADD.D						
1	S.D						
1	DADDIU						
1	BNE						
2	LD.D						
2	DIV.D						
2	LD.D						
2	ADD.D						
2	S.D						
2	DADDIU						
2	BNE						
3	LD.D						
3	DIV.D						
3	LD.D						
3	ADD.D						
3	S.D						
3	DADDIU						
3	BNE						