CSE490/590, Spring 2023, Homework 6 Solution

1. Consider the execution of the following loop, which increments each element of an integer array, on a two-issue processor with speculation

Loop:

LD x2,0(x1) ; x2=array element
ADDIU x2,x2,#1 ; increment x2
SD x2,0(x1) ; store result
ADDI x1,x1,#8 ; increment pointer
BNE x2,x3,LOOP ; branch if not last element

Assume that there are separate integer functional units for effective address calculation, for ALU operations, and for branch condition evaluation. Create a table for the first three iterations of this loop. Assume that up to two instructions of any type can commit per clock (in-order)

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	ld x2,0(x1)	1	2	3	4	5	First issue
1	addi x2,x2,1	1	5		6	7	Wait for ld
1	sd x2,0(x1)	2	3			7	Wait for addi
1	addi x1,x1,8	2	3		4	8	Commit in order
1	bne x2,x3,Loop	3	7			8	Wait for addi
2	ld x2,0(x1)	4	5	6	7	9	No execute delay
2	addi x2,x2,1	4	8		9	10	Wait for ld
2	sd x2,0(x1)	5	6			10	Wait for addi
2	addi x1,x1,8	5	6		7	11	Commit in order
2	bne x2,x3,Loop	6	10			11	Wait for addi
3	ld x2,0(x1)	7	8	9	10	12	Earliest possible
3	addi x2,x2,1	7	11		12	13	Wait for ld
3	sd x2,0(x1)	8	9			13	Wait for addi
3	addi x1,x1,8	8	9		10	14	Executes earlier
3	bne x2,x3,Loop	9	13			14	Wait for addi

CSE490/590, Spring 2023, Homework 6 Solution

2. Suppose we have a VLIW that could issue two memory references, two FP operations, and one integer operation or branch in every clock cycle. Show an unrolled version of the loop

```
Loop:   L.D     F0,0(R1)     ;F0=array element
        ADD.D   F4,F0,F2     ;add scalar in F2
        S.D     F4,0(R1)     ;store result
        DADDUI  R1,R1,#-8    ;decrement pointer
                             ;8 bytes (per DW)
        BNE     R1,R2,Loop   ;branch R1!=R2
```

for such a processor. Unroll seven times to eliminate any stalls. Ignore delayed branches.

Memory reference 1	Memory reference 2	FP operation 1	FP operation 2	Integer operation/branch

Instruction producing result	Instruction using result	Latency in clock cycles
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

Figure 3.2 Latencies of FP operations used in this chapter. The last column is the number of intervening clock cycles needed to avoid a stall. These numbers are similar to the average latencies we would see on an FP unit. The latency of a floating-point load to a store is 0, since the result of the load can be bypassed without stalling the store. We will continue to assume an integer load latency of 1 and an integer ALU operation latency of 0.

Sol) Below Figure.2 shows the code. The loop has been unrolled to make seven copies of the body, which eliminates all stalls (i.e., completely empty issue cycles), and runs in 9 cycles. This code yields a running rate of seven results in 9 cycles, or 1.29 cycles per result, nearly twice as fast as the two-issue superscalar of a 3.2 section called 'Basic Compiler Techniques for Exposing ILP' in the textbook that used unrolled and scheduled code.

Memory reference 1	Memory reference 2	FP operation 1	FP operation 2	Integer operation/branch
L.D F0,0(R1)	L.D F6,-8(R1)			
L.D F10,-16(R1)	L.D F14,-24(R1)			
L.D F18,-32(R1)	L.D F22,-40(R1)	ADD.D F4,F0,F2	ADD.D F8,F6,F2	
L.D F26,-48(R1)		ADD.D F12,F10,F2	ADD.D F16,F14,F2	
		ADD.D F20,F18,F2	ADD.D F24,F22,F2	
S.D F4,0(R1)	S.D F8,-8(R1)	ADD.D F28,F26,F2		
S.D F12,-16(R1)	S.D F16,-24(R1)			DADDUI R1,R1,#-56
S.D F20,24(R1)	S.D F24,16(R1)			
S.D F28,8(R1)				BNE R1,R2,Loop

Figure 2. VLIW instructions that occupy the inner loop and replace the unrolled sequence. This code takes 9cycles assuming no branch delay; normally the branch delay would also need to be scheduled. The issue rate is 23 operations in 9 clock cycles, or 2.5 operations per cycle. The efficiency, the percentage of available slots that contained anoperation, is about 60%. To achieve this issue rate requires a larger number of registersthan MIPS would normally use inthis loop. The VLIW code sequence above requires at least eight FP registers, while the same code sequence for the base MIPS processor can use as few astwo FP registers or as many as five when unrolled and scheduled.