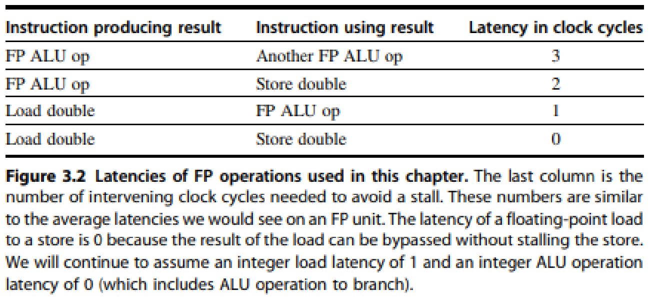
Joe Stanley

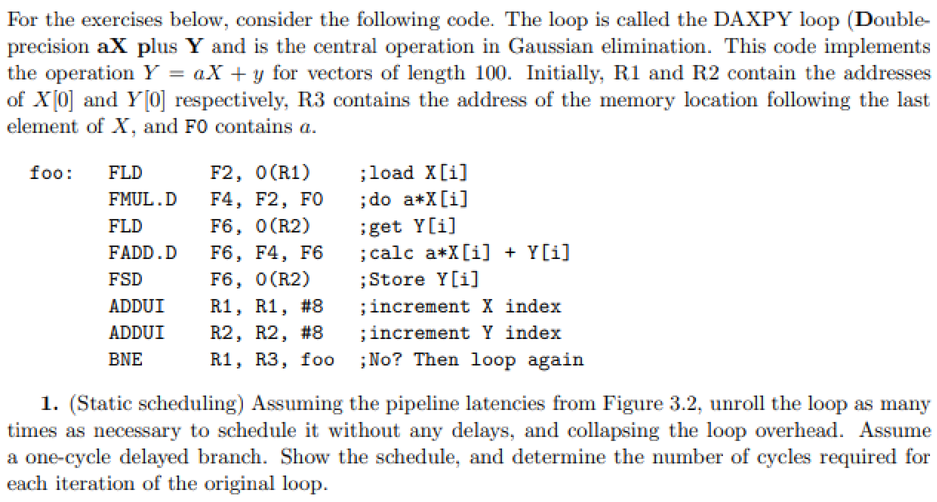
ECE541 - HWK3

Problem 1



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 because 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 (which includes ALU operation to branch).



For the exercises below, consider the following code. The loop is called the DAXPY loop (**D**ouble-precision **aX** **p**lus **Y** and is the central operation in Gaussian elimination. This code implements the operation $Y = aX + y$ for vectors of length 100. Initially, R1 and R2 contain the addresses of $X[0]$ and $Y[0]$ respectively, R3 contains the address of the memory location following the last element of X, and F0 contains a.

```
foo:  FLD     F2, 0(R1)    ;load X[i]
      FMUL.D  F4, F2, F0   ;do a*X[i]
      FLD     F6, 0(R2)    ;get Y[i]
      FADD.D  F6, F4, F6   ;calc a*X[i] + Y[i]
      FSD     F6, 0(R2)    ;Store Y[i]
      ADDUI   R1, R1, #8   ;increment X index
      ADDUI   R2, R2, #8   ;increment Y index
      BNE     R1, R3, foo  ;No? Then loop again
```

1. (Static scheduling) Assuming the pipeline latencies from Figure 3.2, unroll the loop as many times as necessary to schedule it without any delays, and collapsing the loop overhead. Assume a one-cycle delayed branch. Show the schedule, and determine the number of cycles required for each iteration of the original loop.

OP	#	#	#	CYCLE
FLD	F2	0(R1)	-	1
FLD	F6	0(R2)	-	2
FMUL.D	F4	F2	F0	3
ADDUI	R1	R1	#8	4
ADDUI	R2	R2	#8	5
FLD	F2	0(R1)	-	6
FADD.D	F6	F4	F6	7
FMUL.D	F4	F2	F0	8
ADDUI	R1	R1	#8	9
FSD	F6	0(R2)	-	10
FLD	F6	0(R2)	-	11
FLD	F2	0(R1)	-	12
FADD.D	F6	F4	F6	13
FMUL.D	F4	F2	F0	14
ADDUI	R1	R1	#8	15
FSD	F6	0(R2)	-	16
FLD	F6	0(R2)	-	17

Problem 2

2. (Dynamic Scheduling) For this exercise we will add timing to the Tomasulo machine shown in Figure 3.10. Assume the following:

- The number of reservation stations are as shown in Figure 3.10
- There is no forwarding between function units; results are communicated by the CDB.
- One instruction can issue at each clock cycle. Execution of the instruction can begin in the next clock cycle following issue.
- Only 1 value can be on the CDB at a time. If two instructions complete in the same clock cycle, the later instruction is stalled until the next cycle.
- Loads take 1 cycle in execution.

Show the clock cycle when each instruction issues, starts execution, and completes (writes its result onto the CDB) for the first three iterations of the loop. Report your answer in a table similar to Figure 3.23 (Note that figure 3.23 shows a dual issue processor - we are doing a single issue example here.)

ITERATION NUMBER	INSTRUCT	-	-	-	ISSUES AT CLOCK CYCLE NUMBER	EXECUTES AT CLOCK CYCLE NUMBER	MEMORY ACCESS AT CLOCK CYCLE NUMBER	WRITE CDB AT CLOCK CYCLE NUMBER	COMMENT
1	FLD	F2	0(R1)	-	1	2	3	4	FIRST ISSUE
1	FMUL.D	F4	F2	F0	2	5	-	8	WAIT FOR LOAD
1	FLD	F6	0(R2)	-	3	4	5	6	LOAD SECOND
1	FADD.D	F6	F4	F6	4	9	-	-	WAIT 4 MULT
1	FSD	F6	0(R2)	-	5	11	11	-	WAIT 4 ADD
1	ADDUI	R1	R1	#8	6	7	-	-	WAIT FOR LOAD
1	ADDUI	R2	R2	#8	7	8	-	-	WAIT 4 LOAD 2
1	BNE	R1	R3	FOO	8	12	-	-	WAIT 4 ADD
2	FLD	F2	0(R1)	-	9	10	11	12	LOAD 2
2	FMUL.D	F4	F2	F0	10	13	-	16	WAIT FOR LOAD
2	FLD	F6	0(R2)	-	11	14	15	16	LOAD SECOND
2	FADD.D	F6	F4	F6	12	17	-	-	WAIT 4 MULT
2	FSD	F6	0(R2)	-	13	19	19	-	WAIT 4 ADD
2	ADDUI	R1	R1	#8	14	15	-	-	WAIT FOR LOAD
2	ADDUI	R2	R2	#8	15	16	-	-	WAIT 4 LOAD 2
2	BNE	R1	R3	FOO	16	20	-	-	WAIT 4 STORE
3	FLD	F2	0(R1)	-	17	21	22	23	LOAD 3
3	FMUL.D	F4	F2	F0	18	24	-	27	WAIT 4 LOAD
3	FLD	F6	0(R2)	-	19	22	23	24	LOAD SECOND
3	FADD.D	F6	F4	F6	20	28	-	-	WAIT 4 MULT
3	FSD	F6	0(R2)	-	21	30	30	-	WAIT 4 ADD
3	ADDUI	R1	R1	#8	22	25	-	-	WAIT FOR LOAD
3	ADDUI	R2	R2	#8	23	26	-	-	WAIT 4 LOAD 2

ITERATION NUMBER	INSTRUCT	-	-	-	ISSUES AT CLOCK CYCLE NUMBER	EXECUTES AT CLOCK CYCLE NUMBER	MEMORY ACCESS AT CLOCK CYCLE NUMBER	WRITE CDB AT CLOCK CYCLE NUMBER	COMMENT
3	BNE	R1	R3	FOO	24	31	-	-	WAIT 4 STORE

```
In [ ]:
```