CSE490/590, Spring 2023, Homework 6

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)

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

```
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
```

CSE490/590, Spring 2023, Homework 6

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.