6.823 Computer Architecture	DUE: 4/29/2002
Group 18: Timothy Danford, Kilian Pohl, Bryan Russell	
Problem Set 5	*Instructors: Krste Asanovic, Srinivas Devadas*

Problem 3

(a) The main bottleneck for each iteration of the loop are the two vector instructions LV and SV (the other instructions are scalar and can be run in the scalar processor in parallel). Since the vector registers can hold up to 32 elements and there is one lane, we have to read from the register file 32 times. So, in the first two cycles of a given iteration we fetch and decode the LV instruction. In the third cycle we read the first word from memory and in cycle 14 we write back the value to the register file. This continues until we have read the 32 values to the register file. So, the LV instruction completes (last write-back occurs) in cycle 45. During the LV instruction, the SV instruction was fetched and decoded. If we assume no chaining, then the SV instruction can then proceed starting in cycle 46 and finish in cycle 88 (we do not have to worry about dead time for this case since the first element of the vector starts well after cycle 14 when the LV wrote-back the first element plus the 10 cycle dead time). If we assume chaining, then the SV instruction can read the first register starting in cycle 25 (starts after the write-back for the LV instruction in cycle 14 plus the 10 cycle dead-time). Hence, the SV will complete storing all of the registers (last write back) in cycle 67. So, to summarize:

With chaining	67 cycles/32 instr. = 2.09 cycles/instr.
Without chaining	88 cycles/32 instr. = 2.75 cycles/instr.

(b)

```
        MOVI2FP F0,R0           # F0 <- 0
        ADD     R4,R2,R0
        ADD     R5,R1,R0
        ADDI    R7,R0,#32       # R7 <- 32
        MOVI2S  VLR,R7          # VLR <- 32
loop:   LV      V1,R4           # Load next 32 words
        SEQSV   F0,V1           # Check if we have...
        CLZM    R7,VM           # NULL char
        MOVI2S  VLR,R7          # Store number of elts.
                                # to move
        SV      R5,V1           # Store elements
        SLLI    R8,R7,#2
        ADD     R4,R4,R8        # Bump source pointer
        ADD     R5,R5,R8        # Bump destination pointer
```

```
ADDI    R9,R7,#-32      # Check if we have any..
BEQZ    R9,loop         # more to do
```

(c) In this case, the main bottleneck for a given iteration occurs in the sequence of instructions beginning with the LV instruction through the SV instruction (the other instructions can be executed during the SV instruction on the scalar processor). As mentioned in part (a), the LV instruction completely finishes in cycle 45. Now, if we assume no chaining, then the read of the first register for the SEQSV instruction begins in cycle 46 and the write back occurs in cycle 48. So, the entire SEQSV instruction completes in cycle 79. The CLZM and MOVI2S instructions are scalar instructions and can be run on the scalar processor. So, the first register for the SV instruction can finally be read in cycle 84 (the SV instruction's first read stage must wait until the VLR is written back). So, the SV instruction runs until cycle 126. If we assume chaining, then we are able to start the first register of the SEQSV instruction in cycle 25 and write back the first register in cycle 27. So, the SEQSV instruction completely finishes in cycle 58. Since the CLZM and MOVI2S are dependent on the results of the SEQSV instruction (and subsequently the SV instruction is dependent on these instructions), we must wait until the SEQSV instruction completely finishes. So, we are able to start the first read for the SV instruction in cycle 63 and complete the SV instruction in cycle 105. To summarize:

With chaining	105 cycles/32 instr. = 3.28 cycles/instr.
Without chaining	126 cycles/32 instr. = 3.94 cycles/instr.

Comparing with part (a), we see that the memcpy instruction takes fewer cycles to run.