1 Q1 (ISA=30 marks)

1.1 15 marks

Convert to MIPS Assembly, then find the performance (number of cycles) of the following C code assuming:
1-ideal pipelining (no stalls and no start up time)
2- all instructions and data accesses are in cache (100% hit).
3-Processor speed = 1 GHz.

```c
for(i=1; i<=512; i++) {
    Y[i] = Y[i] + A*X[i];
}
```

(Assume A, X[i], Y[i] are 32 bits)

Loop: `Lw R2, 100(C1)`;  `lui R5`

`Lw R3, 5000(C1)`;  `lor R1 Y[i]`

`Mul R4, R2, R6`;  `R6 = A`

`Add R7, R4, R3`

`Sw R7, 5000(C1)`

`Subi R1, 1` #A

`Bnez R1, Loop` [10]

\[ T = 512 \times 7 \times 1 \text{ ns} = 3584 \text{ ns} \]
1.2 15 marks

Convert the above code to MIPS Instructions for Vector Processing, and find the performance in clock cycles assuming the following:
1- MIPS uses a Vector processor with fully pipelined function units and vector registers of 128 elements.
2- MIPS Vector processor runs at 1 GHz.
3- All instructions are in cache (100% hit), data accesses use interleaved memory for load/store and are fully pipelined with no stalls.

```
loop:    
    lov vl, 100(r1) ; v2, 128
    lov v2, 50000(r1) ; y, 128
    mul v3, v1, fo ; x, 128
    addv v4, v3, v2 ; Ax + y, 128
    sv v4, 5000(r1) ;
    subi r1, r1, 4*128;
    bnez r1, loop

T = 4 * (128 + 128 + 128 + 128 + 128 + 2) \times 1 \, nsec
   = 4 \times 642 \, nsec = 2568 \, nsec
```
2 Q2 (Advanced Pipelining= 25 marks)

2.1 scheduling=15 marks

The following code runs in DLX architecture, FP ALU op has latency= 3 cycles, FP ALU op to SD latency= 2 cycles and LD latency = 1 cycle.

```
1- loop:  LD  F0, 0(R1) ; Load X[i]
2-       LD  F4, 0(R2) ; Load Y[i]
3-       ADDD  F6, F4, F0; X[i]+Y[i]
4-       SD  0(R2), F6
5-       SUBI  R1, R1, #8
6-       SUBI  R2, R2, #8
7-       BNEQZ  R5, loop
```

Find the following:

- All types of hazards.

```
RAW
  1 → 3  j  2 → 3, 3 → 4
      5-    7
```

CONTROL  BNEQZ

- The performance of the above (cycles per loop).

```
loop:  LD  F0, 0(C(R1))
  1  LD  F4, 0(C(R2))
  2  ADD  F6, F4, F0
  3  SD  0(C(R2)), F6
  4  SUBE  R1, 121, #8
  5  SUBI  R2, 121, #8
  6  BNEQZ  R5, loop
  7  small  Control  12 cycles / loop
```

12 cycles / loop
• Use loop unrolling 3 times and schedule the code to improve performance.
  Find the performance of the code.

LOOP:  

load constant 0 (R1)  
load constant -8 (R1)  
load F3, -16 (R1)  
load F4, 0 (R2)  
load F8, -8 (R2)  
load F16, -16 (R2)  
add F6, F4, F0  
add F12, F8, F1  
add F14, F3, F10  
set 0 (R2), F6  
subtract R15, R1, #24  
subtract R2, R2, #12  
set +16 (R2), F12  
bnez R17 loop  

cycles/loop = \frac{15}{3} = 5 \text{ cycles}
2.2 multiple-issue=10 marks

In the above example of 2.1, assume that the DLX uses a superscalar to issue two instructions on each clock (1 Int, 1 FP). Write the code for the superscalar and find the performance in cycles per loop.

<table>
<thead>
<tr>
<th>Loop</th>
<th>Int 2</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LD F0, 0(CR1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>LD F1, -0(CR1)</td>
<td>ADD F6, F1, F0</td>
</tr>
<tr>
<td>3</td>
<td>LD F8, -8(CR2)</td>
<td>ADD F6, F1, F0</td>
</tr>
<tr>
<td>4</td>
<td>LD F3, -16(CR1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LD F10, -16(CR2)</td>
<td>ADD F12, F8, F1</td>
</tr>
<tr>
<td>6</td>
<td>LD F6, 0(CR2)</td>
<td>ADD F14, F3, F6</td>
</tr>
<tr>
<td>7</td>
<td>SUB R2, R2, #24</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SUB R2, R2, #24</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SUB R2, R2, #24</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>LD F12, -16(CR2), F12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BR NZ R4, warp</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LD F14</td>
<td></td>
</tr>
</tbody>
</table>

Performance = \( \frac{12}{3} = 4 \) cycle / loop
Q3 (Advanced Pipelining = 15 marks)

The following code runs in a DLX architecture with scoreboard, FP ADD op has latency = 2 cycles, FP MULTD latency = 8 cycles, LD latency = 1 cycle, and 2 ADD/SUB function units.

LD F2, 0(R1) ; Load X[i]
MultD F4, F0, F2 ; aX[i]
ADD D F10, F4, F8
SUBD F8, F8, F0
SD F8, 0(R1)

- Find the Scoreboard instruction status at the end of code execution.
4. Q4 (Multiprocessor Systems= 30 marks)

4.1 =4

Compare the advantages and disadvantages of write invalidate versus write update for multiprocessor systems.

write invalidate: uses bus only once, (bus utilization is better) write updates
write update: uses bus each time, needs to update, sending data on bus, but has faster read after write update (fast read)

4.2 =3

Explain how does multiprocessor system with two level cache maintain coherency.

- Inclusion, each block of C1 is in C2.
- Only C2 snoops on bus, if block is not in C1, need to invalidate it in C1, but if it is in C1, needs validation.
4.3  =8

Assume a shared memory multiprocessor system that uses the write invalidate snooping coherency protocol. Find the state of the cache block after each of the following operations:

- Processor read miss to a private block in its cache
  \[\text{shared}\]

- Processor write hit to invalid block in its cache
  \[\text{private}\]

- Processor write miss to shared block in its cache
  \[\text{private}\]

- Bus read operation hit to a private block in cache
  \[\text{shared}\]
4.4  =15

In a shared memory multiprocessor system assume the following:-
1-the system uses the write invalidate coherency protocol.
2-All processor’s speed = 1 GHz. Processor P1 has R1= 100 and Processor P2
has R3= 200. All other registers =0.
3-Each processor uses a direct mapped cache. The cache size = 512 Kbytes,
block size= 8 bytes. The cache speed is the same as processor speed.
4-The bus width is 8 bytes and bus latency and waiting time = 200 ns. Memory
latency = 100 ns.

The above system is executing the following events:-

<table>
<thead>
<tr>
<th>step#</th>
<th>P1</th>
<th>P2</th>
<th>OUTCOME, TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LW R4, 100(R1)</td>
<td></td>
<td>read miss, shared C1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>LW R5, 104(R1)</td>
<td>read miss, shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From C1 = 200+1+1 = 202</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SW 100(R1), R7</td>
<td></td>
<td>write hit,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private, invalidate C2 T = 200+1+1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>LW R16,100(R1)</td>
<td>newly shared cop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shared C4 T = 200+1+1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LW R12, 100(R1)</td>
<td></td>
<td>miss 2 hit = 1 cycle</td>
</tr>
</tbody>
</table>
<pre><code>   |             | Shared C1 + C2               |
</code></pre>

Find the outcome and time required to perform each event in the above code (state, processor, bus, memory operations and cost in time).