EECSE 469 Computer Architecture I
Brayden Lam, Christopher Andrade
HW 4

1. For the following snippets of assembly, determine the type of the hazard(s), which instruction(s), and which register(s) it occurs for. Assume the forwarding logic has been implemented, and the branch misprediction penalty is 4 cycles. Assume there is code above and below the provided code. Each part of this question is independent from the other parts.

```
Example:
AND R3, R5, R7
ADD R6, R3, R7   // Data Hazard, R3 (Read after Write)
```

a.
```
ADD R1, R2, R3
SUB R8, R1, R2   // Data Hazard, R1 (Read after Write)
AND R3, R3, R2
```

b.
```
ADD R1, R2, R3
STR R1, [R2, #0] // Data Hazard, R1 (Store after Write)
SUB R2, R1, R2   // Data Hazard, R1 (Read after Write)
LDR R3, [R2, #0] // Data Hazard, R2 (Read after Write)
```

c.
```
      LSL R1, R3, #2
      ADD R1, R1, R4   // Data Hazard, R1 (Read after Write)
      B END            // Control Hazard, Branches
      LSL R1, R4, #2
      ADD R1, R2, R5
END
```

d.
```
[For this question, Let R1 = 0, R2 = 7, R4 = 3]
START
      ADD R3, R2, R4
      AND R4, R2, R4
      CMP R3, R4       // Data Hazard, R3, R4 (Read after Write)
      BLT START        // Control Hazard, Unsuccessful Branch
      SUB R0, R4, R3
      STR R0, [R2, #8]// Data Hazard, R0 (Store after Write)
```

2. The following snippets of assembly include data hazards. Indicate where to insert no-ops and how many, or which instructions to stall, in order for this code to run on the 5-stage processor discussed in class. Assume no forwarding, and the register file is written to on the falling edge. Assume there is code above and below the provided code. Each part of this question is independent from the other parts.

a.
```
AND R0, R1, R3
// 2 no-ops
ADD R1, R2, R0
SUB R7, R8, R9
// 1 no-op
ORR R3, R1, R8
```

b.
```
AND R0, R1, R3
LDR R1, [R2, #0]
ORR R1, R3, R8
// 2 no-ops
LDR R2, [R1, #0]
AND R1, R3, R6
ORR R2, R3, #6
```

3. The following snippet of assembly includes control hazards:

```
BEGIN
      LSR R2, R3, #4
      SUB R1, R1, R4
      B END
      LSL R1, R4, #2
      ADD R1, R2, R5
END
      AND R1, R3, R6
      ADD R7, R2, R4
      MOV R0, #100
      B BEGIN
      LDR R9, [R0, #0]
      LSL R0, #3
      ORR R8, R8, R7
      MOV R5, R9
```

a. Indicate which instructions will need to be flushed for the B END and B BEGIN instructions, and what the next instruction executed will be. Assume that this code is run on the 5-stage processor discussed in class. Assume no forwarding, and the register file is written to on the falling edge. Assume that there is no early branch target resolution. Assume there is code above and below the provided code. Each part of this question is independent from the other parts.

For B End:

Flushed:

```
LSL R1, R4, #2
ADD R1, R2, R5
```

Next instruction:

```
AND R1, R3, R6
```

For B begin:

Flushed:

```
LDR R9, [R0, #0]
LSL R0, #3
ORR R8, R8, R7
MOV R5, R9
```

Next instruction:

```
LSR R2, R3, #4
```

b. Repeat part a, but this time, assume we have implemented the logic to support early branch resolution.

For B End:

Flushed:

```
LSL R1, R4, #2
ADD R1, R2, R5
```

Next instruction:

```
AND R1, R3, R6
```

For B begin:

Flushed:

```
LDR R9, [R0, #0]
LSL R0, #3
```

Next instruction:

```
LSR R2, R3, #4
```