Computer Architecture

Exercises: Instruction Pipeline

Task 1. Consider the following sequence of instructions.

```
i1:  R0 ← MEM [R1+8]
i2:  R2 ← R0 * 3
i3:  R3 ← MEM [R1+12]
i4:  R4 ← R3 * 5
i5:  R0 ← R2 + R4
```

These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

(a) Identify all the data dependencies among the instructions

(b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:

- D^*: the reason of the stall is data hazard
- S^*: the reason of the stall is structural hazard
- C^*: the reason of the stall is control hazard

(c) For each instruction decide whether *forwarding* is necessary in the EX stage, if yes, tell which pipeline register the operand is read from

(d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

Task 2. Consider the following sequence of instructions.

```
i1:  R5 ← MEM [R3+24]
i2:  R6 ← MEM [R4+16]
i3:  R7 ← R6 + R5
i4:  R8 ← R6 - R5
i5:  R5 ← R7 * R8
i6:  R4 ← R4 + 4
```

These instructions are executed on a CPU which uses a 6-stage pipeline. The execution of the instructions consists of 5 phases: fetch (IF), decode (ID), execute arithmetic-logic operations (EX), perform memory operations (MEM), and writing back the result into the register file (WB). The latency of the ID, EX, MEM, WB phases is 1 clock cycle, while the latency of the IF phase is 2 clock cycles, the iteration interval is still 1. This means that the instruction fetch phase consists of two stages: IF0 and IF1.

(a) Identify all the data dependencies among the instructions

(b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:

- D^*: the reason of the stall is data hazard
- S^*: the reason of the stall is structural hazard
- C^*: the reason of the stall is control hazard

(c) For each instruction decide whether *forwarding* is necessary in the EX stage, if yes, tell which pipeline register the operand is read from

(d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

Task 3. Consider the following sequence of instructions.

```
i1:  D0 ← D1 * D2
i2:  D3 ← D0 + D5
i3:  MEM [R0+4] ← D3
i4:  MEM [R0+12] ← D0
```

The D0-D5 registers store floating point numbers, the R0 stores integer number.

These instructions are executed on a CPU which uses a 6-stage pipeline. The execution of the instructions consists of 5 phases: fetch (IF), decode (ID), execute arithmetic-logic operations, perform memory operations (MEM), and writing back the result into the register file (WB). An arithmetic operation can be an integer operation (EX, latency is 1), a floating point addition operation (latency is 4, iteration interval is 1, A0, A1, A2, A3) or a floating point multiplication operation (latency is 7, iteration interval is 1, M0, M1, M2, M3, M4, M5, M6). These three operations can be performed in parallel, if there are no hazards.

(a) Identify all the data dependencies among the instructions

(b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:

- D^*: the reason of the stall is data hazard
- S^*: the reason of the stall is structural hazard
- C^*: the reason of the stall is control hazard

(c) For each instruction decide whether *forwarding* is necessary in the EX stage, if yes, tell which pipeline register the operand is read from

(d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

Task 4. Consider the following sequence of instructions.

```
i1:  D2 ← D0 * D1
i2:  MEM [R0+0] ← D2
i3:  D2 ← D0 + D1
i4:  MEM [R0+8] ← D2
```

The pipeline is the same as in the previous example.

(a) Identify all the data dependencies among the instructions

(b) Schedule the instructions in the pipeline