Dependency Analysis of Instructions

17PW24 - Niteshkumar S

Dependence analysis

Dependence analysis produces execution-order constraints between statements / instructions. Broadly speaking, a statement S2 depends on S1 if S1 must be executed before S2. Dependence analysis determines whether it is safe to reorder or parallelize statements. Broadly, there are three types of dependencies,

- Control Dependencies.
- Data Dependencies.
- Name Dependencies.

Control Dependency

Control dependency is a situation in which a program instruction executes if the previous instruction evaluates in a way that allows its execution.

```
# Begin

if(x > 10){         // s1
        y = 20  // s2
}
y = 30              //s3

# End
```

Here statement S2 is conditionally guarded by S1, Where S2 gets executed only if S1 turns out to be true. Hence S2 has control dependency over S1

Data Dependency

A data dependency is a situation in which a program statement (instruction) refers to the data of a preceding statement.

Flow dependency (True dependency)

A Flow dependency, also known as a data dependency or true dependency or read-after-write (RAW), occurs when an instruction depends on the result of a previous instruction

```
# Begin
A = 3   // S1
B = A   // S2
C = B   // S3
# End
```

Here, Instruction 3 is truly dependent on instruction 2, as the final value of C depends on the instruction updating B. Instruction 2 is truly dependent on instruction 1, as the final value of B depends on the instruction updating A. Instruction level parallelism is therefore not an option in this example.

Anti-dependency

A statement S2 is anti dependent on S1 if and only if S2 modifies a resource that S1 reads and S1 precedes S2 in execution. The following is an example of an anti dependence (WAR: Write After Read). Here, S2 sets the value of y but S1 reads a prior value of y.

```
# Begin
S1      x = y + c
S2      y = 10
# End
```

Output dependency

An output dependency, also known as write-after-write (WAW), occurs when the ordering of instructions will affect the final output value of a variable.

```
# Begin
S1      B = 3
S2      A = B + 1
S3      B = 7
# End
```

In the example below, there is an output dependency between instructions 3 and 1 — changing the ordering of instructions in this example will change the final value of A, thus these instructions cannot be executed in parallel.

Name Dependency

The name dependency occurs when 2 registers use the same register memory location, called a name, but there is no flow of data between instructions associated with the name.

There are 2 types of name dependencies between instruction I and j

1. An anti dependence between instruction I and instruction j occurs when instruction j writes a register or memory location that instruction I reads. The original ordering must be preserved to ensure that I read the correct value.
2. An output dependence occurs when instruction i and instruction j write the same register or memory location. The ordering between the instructions must be preserved to ensure that the value finally written corresponds to instruction j.