Dependency Analysis of Instructions

Aakash Hemadri <17pw01@psgtech.ac.in>

Dependence Analysis

A data dependency in computer science is a situation in which a program statement (instruction) refers to the data of a preceding statement. In compiler theory, the technique used to discover data dependencies among statements (or instructions) is called dependence analysis.

There are three main types of dependencies: data, name and control.

Data dependency:

- `Flow Dependency:` Occurs when an instruction cannot proceed without the result of previous instruction, read-after-write (RAW)

```
1. A = 3
2. B = A # Depends on 1
3. C = B # Depends on 2
# These three atomic instructions may not be parallelized
# Instruction level parallelism is not an option here.
```

- `Anti Dependency:` An anti-dependency, also known as write-after-read (WAR), occurs when an instruction requires a value that is later updated.

```
1. B = 3
2. A = B + 1
3. B = 7
```

- `Ouput 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.

```
1. B = 3
2. A = B + 1
3. B = 7
```

Name Dependency:

- Occurs as a consequence of incomplete or waiting of a register to load an address used in further indirect memory addressing which will delay further retrieval of operands.
- Anti Dependency & Output Dependency are Name dependencies – it can be removed by renaming of variables/registers

Control Dependency:

- This type of dependency occurs during the transfer of control instructions such as BRANCH, CALL, JMP, etc. On many instruction architectures, the processor will not know the target address of these instructions when it needs to insert the new instruction into the pipeline. As a result instructions read into pipeline will have to be flushed.

Minimizing Performance Degradation

- Conditional branching can be resolved with prefetching target instruction.

- Using branch target buffer, it is an associative memory accompanied with the fetch segment of the pipeline, it holds the addresses of previously executed branch instructions and targets as well. This allows pipeline to quickly retrieve instructions through the associative memory BTB, if available it continues prefetching from the new path.
- Variation of BTB is the loop buffer, a smaller albeit faster register file maintained by the instruction fetch segment og the pipeline, Used to hold loops when detected in the program.
- Branch predicition is said to be the most impactful to increasing pipeline practical speeds in early iterations of the x86 architecture. This method uses additional logic to guess the outcome of a conditional branch instruction before it is executed. The pipeline then begins prefetching the instructions from the predicted path. A correct prediction eliminates the wasted time caused by branch prediction penalties.
- Delayed branching, employed by RISC processors during compilation by rearranging the machine language code sequence by inserting useful instructions like NOOP instruction allowing a continous flow of the pipeline.