Predication and Speculation

Compilers for High Performance Computers

Stefano Petrilli
stefano.petrilli@upc.edu

Jakob Eberhardt
jakob.eberhardt@estudiantat.upc.edu

November 17, 2024

Problem Introduction

```
1 void process_data(int *data, int size) {
2     for (int i = 0; i < size; i++) {
3         if (data[i] < 50) {
4             data[i] = 0;
5         }
6     }
7 }
```

Listing: For `size=100000000`, we take 1.348 seconds

Conditional Codes

- Concern control flow of the program
- Jump to a different instruction based on a condition
- May effect performance

Instructions have to wait for dependencies

```
1  loop:
2      cmp w21, w20                    // compare i with size
3      b.ge end_loop                   // if i >= size, exit loop
4      ldr w0, [x19, w21, UXTW #2]     // w0 = data[i]
5      cmp w0, #50
6      b.ge skip_update                // if data[i] >= 50, skip
7      mov w0, #0                      // w0 = 0
8      str w0, [x19, w21, UXTW #2]     // data[i] = w0
9  skip_update:
10     add w21, w21, #1                // i++
11     b loop
```

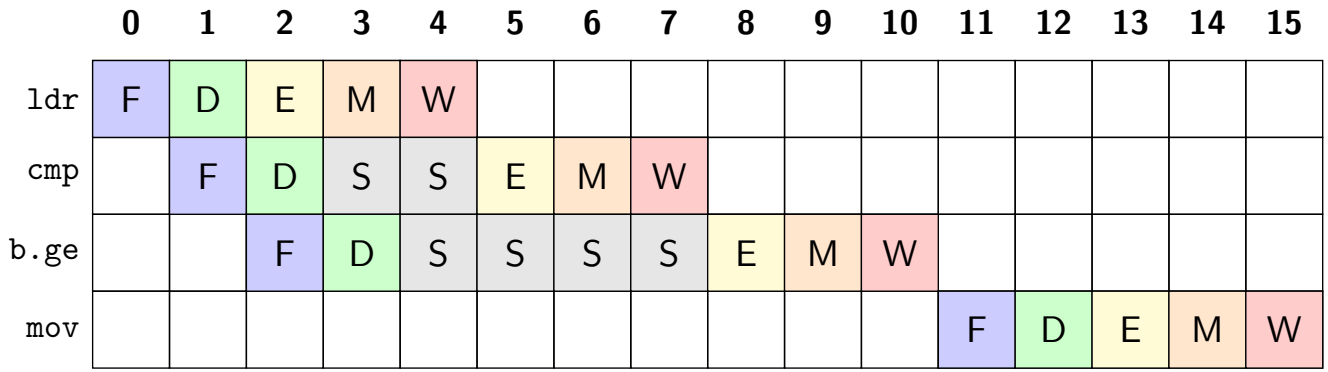


Figure: We need to wait for `data[i]` to be written back to `w0` for the comparison. We also need to wait for `b.ge` to finish to know which instruction to fetch next

Instructions have to wait for dependencies

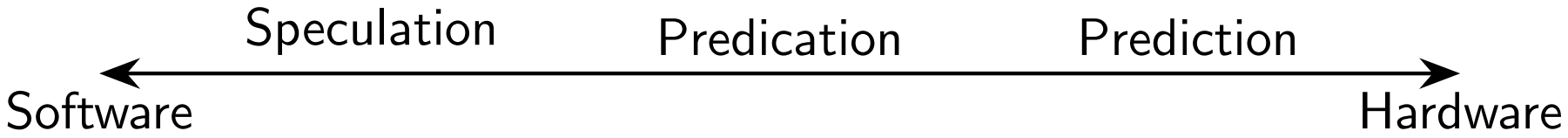
Problem

- We have to wait for `data[i]` to be loaded
- We cannot exploit available resources
- We do not know which instruction to fetch next
- Pipeline stalls, IPC↓
- **We need to reduce the cost of conditional branching**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
`ldr`	F	D	E	M	M	M	M	M	W											
`cmp`		F	D	S	S	S	S	S	S	E	M	W								
`b.ge`			F	D	S	S	S	S	S	S	S	S	E	M	W					
`mov`																F	D	E	M	W

Figure: A Cache miss may effect the pipeline even more.

Methods



- **Speculation**
  - Speculate about control flow at run time while compiling
  - Generate additional code, e.g., compensation code
- **Predication**
  - Execute multiple branches
  - Select correct result when condition arrives
  - Requires special instructions, e.g., `cmov`
- **Prediction**
  - Try to predict control flow during run time
  - Using a hardware structure such as a branch predictor

Speculation

Definition

"An instruction is *speculatively* executed if it is moved above a conditional branch that it is control dependent upon" [1]

- **Compiler-controlled**
  - Speculate about real control flow
  - Try to enable other optimization opportunities
- **Control Speculation**
  - Execute instructions before control flow was determined
- **Data Speculation**
  - Execute instructions with potentially wrong data

Exceptions

Type	Behavior	Example
safe	Never causes an exception to a dominating block	Register-to-register operations
unsafe	Can always cause an exception to a dominating block	Loads, certain floating-point instructions

Predicated Execution without Branching

```
1  loop_predicated:
2      cmp w21, w20                    // compare i with size
3      b.ge end_predicated             // if i >= size, exit
           loop
4
5      ldr w0, [x19, w21, UXTW #2]     // w0 = data[i]
6      cmp w0, #50
7      csel w0, wzr, w0, lt            // if w0 < 50, w0 = 0;
           else w0 = w0
8      str w0, [x19, w21, UXTW #2]     // data[i] = w0
9
10     add w21, w21, #1                // i++
11     b loop_predicated
```

Speculative Execution with Branching

```
1  loop_speculative:
2      cmp w21, w20                    // compare i with size
3      b.ge end_speculative            // if i >= size, exit loop
4
5      ldr w0, [x19, w21, UXTW #2] // w0 = data[i]
6      cmp w0, #50
7      b.ge skip_update                // if data[i] >= 50, skip
           update
8      mov w0, #0                      // w0 = 0
9      str w0, [x19, w21, UXTW #2] // data[i] = w0
10 skip_update:
11     add w21, w21, #1                // i++
12     b loop_speculative
```

References & Questions I

[1] P.P. Chang et al. “Three architectural models for compiler-controlled speculative execution”. In: *IEEE Transactions on Computers* 44.4 (1995), pp. 481–494. DOI: 10.1109/12.376164.