3rd　Delay Branch and Dynamic Scheduling

（Ver. 0.9 2018-4-24 By Wangxp@fudan.edu.cn)

1. Objective

1.1 Familiar with delay branch;

1.2 Familiar with the code moving when dealing with delay branch;

1.3 Familiar with loop unrolled;

1.4 Familiar with scoreboarding and tomasulo algorithm;

1.5 Familiar with branch performance.

2. Files & Document

Code for DLX, not scheduled for the pipeline: (from **Hennessey and Patterson**)

```
Loop:
        LD          F0, 0(R1)     ; F0=array element
        ADDD        F4, F0, F2     ; add scalar in F2
        SD          0(R1), F4      ; store result
        SUBI        R1, R1, #8     ; decrement pointer 8 bytes per DW
        BNEZ        R1, Loop       ; iterate if R1 not zero.
end.
```

Now the unrolled loop after it has been rescheduled: (Assume we will unroll it 4 times)

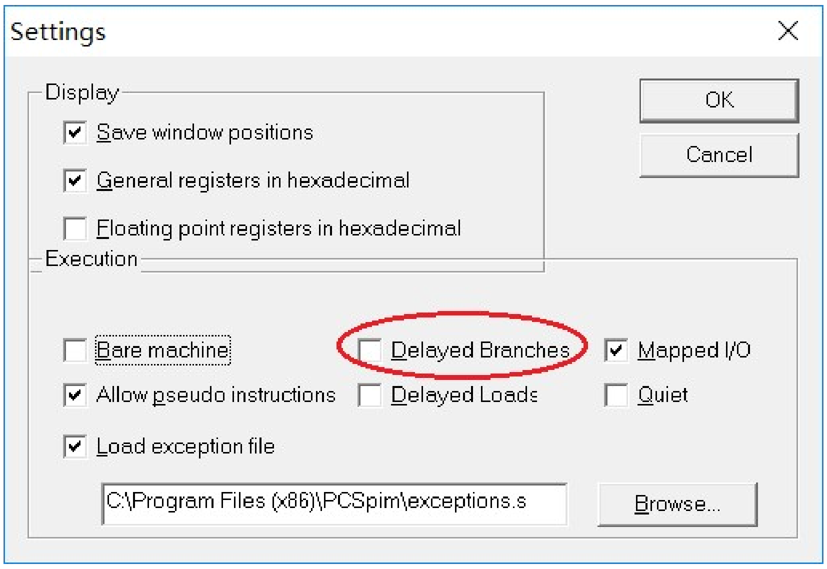
```
Loop:
        LD          F0, 0(R1)
        LD          F6, -8(R1)
        LD          F10, -16(R1)
        LD          F14, -24(R1)
        ADDD        F4, F0, F2
        ADDD        F8, F6, F2
        ADDD        F12, F10, F2
        ADDD        F16, F14, F2
        SD          0(R1), F4
        SD          -8(R1), F8
        SD          -16(R1), F12           ; You might be tempted to do the next
                                           ; SD at this point, but it is necessary
                                           ; for the branch delay slot to avoid stalls
        SUBI        R1, R1, #32            ; Here you update the counter
        BNEZ        R1, Loop
        SD          8(R1), F16               ; 8-32 = -24  Branch Delay Slot
```

You save the "SD 8(R1), F16" instruction for after the branch because it legally fills the requirements. It would have been executed whether the branch was taken or not, so it is independent of the branch instruction.

3. Tasks

3.1 Use PCSPIM to observe the Delayed Branches behavior.

You can set Delayed Branches option via **Simulator--> settings--> Delayed Branches**(**Execution Section**). See the diagram below, with red circles.



Settings

Display

- ✓ Save window positions
- ✓ General registers in hexadecimal
- ☐ Floating point registers in hexadecimal

Execution

- ☐ Bare machine
- ☐ Delayed Branches
- ✓ Mapped I/O
- ✓ Allow pseudo instructions
- ☐ Delayed Loads
- ☐ Quiet
- ✓ Load exception file

C:\Program Files (x86)\PCSpim\exceptions.s

Understand the code in Section 2, and then place the code in PCSpim to run (You should add whatever to make the program to run), and make comparison when setting Delayed Branches.

3.2 Understand scoreboarding and tomasulo algorithm

Refer [3][4] to understand scoreboard & tomasulo algorithms, esp. for load & store operations.

3.3 Understand branch performance problem

Refer [2][5] to understand branch performance problem. And answer following problems.

3.3.1 Why the following codes (code 3.1 and Code 3.2) are equivalent?

Give: `int data[arraySize];`

if (data[c] >= 128)
  sum += data[c];
	int t = (data[c] - 128) >> 31;
	sum += ~t & data[c];

Code 3.1	Code 3.2

3.3.2 * Write or add C++ code to compare performance of different schemes

- Primary Requirement

Using c++ programming language;

Using a large array (size=32768);

A sample program called **parray.cpp** is attached.

- Schemes

Scheme	Scheme name	Description
1	Random	All data are generated using random lib of C++.
2	Sorted	All data are generated using random lib of C++. Before processing, data are sorted.
3	Random-hacked	As in [1].
4	Sorted-hacked	As in [3].

You should fill your results into the Table below. And you can design your own table to record your performance procedure.

Scheme	Scheme name	Compiler and option	AVG time	Rank
1	Random			
2	Sorted			
3	Random-hacked			
4	Sorted-hacked			

4. References

[1] Branch Delay Slot. https://en.wikipedia.org/wiki/Delay_slot

[2] Branch Predictor. https://en.wikipedia.org/wiki/Branch_predictor

[3] Scoreboarding Algorithm. https://en.wikipedia.org/wiki/Scoreboarding

[4] Tomasulo Algorithm. https://en.wikipedia.org/wiki/Tomasulo_algorithm

[5] Why sorted array preferred?. https://stackoverflow.com/questions/11227809/why-is-it-faster-to-process-a-sorted-array-than-an-unsorted-array#11227902