Stage1

Main functionality: instruction decoding and issuing

Input: 2 instructions, flush & bid, writeback information

Format:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31-28 | 27-24 | 23-20 | 19-16 | 15-11 | 10-0 |
| Opcode | Destination | Source1 | Source2 | immediate | Reserved bits |

Add instruction, imm == 0, opcode == 1000

Load instruction, source2 == 0, opcode == 0100

Store instruction, des == 0, opcode == 0010

Branch instruction, des == 0, opcode == 0001

5-bit immediate indicates the size of icache/datacache is 32.

Flush & bid are sent back from branch\_ctrl

Write back information comes from the last stage of our pipeline design. It is used to update the raw\_history.

Five units are included in this stage:

1. Decode

Pure combinational, decode the instructions.

Input: two instructions every cycle in the format shown above.

Output: ins1\_op, ins1\_des, ins1\_s1,ins1\_s2, ins1\_imm and same for ins2.

1. Issue queue

It has four basic functionalities: new instruction allocation, instruction issuing and deallocation, shift and flush.

Its inputs come from two parts: decode and precalculation.

It has 16 entries. When it’s full, a full signal will be set to high and it will go down to 0 only after it’s empty. This signal is used to control the pc. (Notice: when there is only 1 or 0 empty entry, full signal will be high.)

Every instruction that comes into this queue will be assigned to a branchID, which is 3-bit wide. When the branchID is used up, branch\_full signal will be high. It will go down only when the queue is empty again.

1. All\_checker

It consists of five submodules:

raw\_history\_check: It keeps track of the usage of each register and checks if new instructions will read from any register which will be written in the next few cycles. If so, those instructions can’t be issued.

load\_store\_check: It makes sure that every cycle, at most 1 memory instruction can be issued and instructions after the second valid memory instruction (which can’t be issued) can’t be issued out of order.

raw\_war\_checker: it checks the data dependency among the four instructions

branch\_check: it makes sure that branches won’t be issued out of order

1. ins\_swap:

If a memory instruction should be issued, it has to stay in the “first” place because the first ALU will be used for effective address calculation. Therefore, after all\_checker, the four instructions (now all potential hazards are eliminated) will be sent into this unit for “reordering”

1. Precalculation

It consists of three submodules: entry\_select, shift\_amount, and new\_ins\_addr\_calculation

Entry\_select

Since in each cycle, the number of instructions that can be issued varies, and only at the positive edge of the next cycle can we know the index of the instructions that are actually issued. Therefore, this module is used to choose the index of instructions that should be sent out from the issue queue.

Shift\_amount

The shift function of the queue is realized by the help of this module which calculate the number of entries the instructions stored in the issue queue should shift up.

new\_ins\_addr\_calculation

Calculate the index for new instruction allocation.