B1) Let us consider a pipeline with the following in-order stages: Fetch (IF), and Decode (DEC), and an out-of-order, dynamic speculative execution unit, for integer, floating point, and memory access instructions. The Commit (CO) phase is carried out in program order.

Speculation is carried out through a ROB and a number of Reservation Stations: the ROB has 8 entries. There is a single reservation station for each of the following Functional Units:

* Int1 for arithmetic, logical, address computation, branch and jump
* Fadd1 floating point add/subtraction
* Fadd2 floating point add/subtraction
* Fmolt1 floating point and integer multiplication
* Fmolt2 floating point and integer multiplication
* Fdiv1 floating point division

Load and store instructions are carried out through special purpose buffers:

* Load1
* Load2
* Store1
* Store2

The execution model for the dynamically scheduled out-order unit is described in an associated state transition diagram. All functional units are blocking (no internal pipelining).

The ISA supports 28 integer registers (R1-R28) and 28 floating point registers (F0-F27). All instructions have a 4-byte format.

The memory hierarchy consists of the following caches:

L1 D-cache, 2-way, 16 KB, block length 16 bytes;

L1 I-cache, direct mapped, 8 KB, block length 8 bytes;

L2 unified cache, 4-way, 512 KB, block length 32 bytes.

Caches are pipelined, non blocking. Latencies and hit/miss times are as follows:

|  |  |
| --- | --- |
| Int - 1 | Hit L1 – 1 |
| Fmolt – 4 | Hit L2 – 3 |
| Fadd - 2 | Miss L2 – 6 |
| Fdiv - 5 |  |

Further assumptions:

a) no BP, no BTB;

b) the code of B2) is already loaded in the caches;

c) L1 D-cache is empty

d) hit/miss costs in the hierarchy are incremental: a hit in L2 has an overall cost of 1+3 clock cycles.

B2) The following code executes operations on arrays X() and Y(), both 100 elements of 8-byte floating point data. X() is allocated from 0[R1], Y() from 0[R2], and [R3]=100 at the beginning.

**PC1** Loop LD F1,0(R1) ; loads X(i)

**PC2**  SUBI R3,R3,1

**PC3**  ADDI R1,R1,8

**PC4**  ADDF F3,F2,F1 ; F2 has a values computed before Loop

**PC5**  LD F4,0(R2) ; loads Y(i)

**PC6**  SD -16(R1),F3

**PC7**  MULTF F5,F1,F2

**PC8**  ADDF F6,F1,F4

**PC9**  MULTF F5,F6,F5

**PC10** SD 0(R2),F6

**PC11** SD 512(R2),F5

**PC12** ADDI R2,R2,8

**PC13** BNEZ R3,Ciclo

1) Enumerate the timing of the state transitions for each instruction in the first 2 iterations (highlighting all conflicts). Assume the time of issue for PC1 is 1.

2) Show the contents of the reservation stations at the issue time of PC9.

3) Show the ROB’s content at the issue time of PC13 in the first iteration (specify the head and tail positions within the ROB).

Dynamically scheduled, speculative execution – Tomasulo’s algorithm
data structures

De-coupled ROB RS model

|  |  |  |
| --- | --- | --- |
| **INSTRUCTION** |  | **STATE** |
|  | **ROB** | **WO** | **RE** | **DI** | **EX** | **WB** | **RR** | **CO** |
| **PC1**  LD F1,0(R1) | **0** |  | **1** | **2** | **3-12** | **13** | **14** | **15** |
| **PC2**  SUBI R3,R3,1 | **1** |  | **2** | **3** | **4** | **5** | **6-15** | **16** |
| **PC3**  ADDI R1,R1,8 | **2** |  | **3** | **4-5** | **6** | **7** | **8-16** | **17** |
| **PC4**  ADDF F3,F2,F1 | **3** | **4-13** | **14** | **15** | **16-17** | **18** | **19** | **20** |
| **PC5**  LD F4,0(R2) | **4** |  | **5** | **6** | **7-16** | **17** | **18-20** | **21** |
| **PC6**  SD -16(R1),F3 | **5** | **6-18** | **19** | **20** | **21-30** | **-** | **31** | **32** |
| **PC7**  MULTF F5,F1,F2 | **6** | **7-13** | **14** | **15** | **16-19** | **20** | **21-32** | **33** |
| **PC8**  ADDF F6,F1,F4 | **7** | **8-17** | **18** | **19** | **20-21** | **22** | **23-33** | **34** |
| **PC9** MULTF F5,F6,F5 | **0** | **16-22** | **23** | **24** | **25-28** | **29** | **30-34** | **35** |
| **PC10** SD 0(R2),F6 | **1** | **17-22** | **23** | **24** | **25** | **-** | **26-35** | **36** |
| **PC11** SD 512(R2),F5 | **2** | **18-29** | **30** | **31** | **32-41** | **-** | **42** | **43** |
| **PC12** ADDI R2,R2,8 | **3** |  | **21** | **22** | **23** | **24** | **25-43** | **44** |
| **PC13** BNEZ R3,PC1 | **4** |  | **22-23** | **24** | **25** | **-** | **26-44** | **45** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
|  | Reservation Stations (RS) |
| Busy | Op | Vj | Vk | ROBj | ROBk | Rob entry | Ind |
| Int1 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| FAdd1 |  |  |  |  |  |  |  |  |
| FAdd2 |  |  |  |  |  |  |  |  |
| FMolt1 |  |  |  |  |  |  |  |  |
| FMolt2 |  |  |  |  |  |  |  |  |
| FDiv1 |  |  |  |  |  |  |  |  |
| Load1 |  |  |  |  |  |  |  |  |
| Load2 |  |  |  |  |  |  |  |  |
| Store1 |  |  |  |  |  |  |  |  |
| Store2 |  |  |  |  |  |  |  |  |

ROBj ROBk: sources of operands not yet available

ROB entry: position in the ROB of the instruction

|  |  |
| --- | --- |
|  | Result Register status |
| Int | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 |
| ROB # |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| status |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Int | R15 | R16 | R17 | R18 | R19 | R20 | R21 | R22 | R23 | R24 | R25 | R26 | R27 | R28 |
| ROB # |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| status |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 |
| ROB # |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| status |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 | F22 | F23 | F24 | F25 | F26 | F27 |
| ROB # |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| status |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |
| --- |
| Reorder Buffer (ROB) |
| Entry # | Busy | Op | State | Destination | Value |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |

WO

RE

DI

EX

WB

RR

CO

queue

 **Decoupled execution model**

The state diagram depicts the model for a dynamically scheduled, speculative execution microarchitecture equipped with a Reorder Buffer (ROB) and a set of Reservation Stations (RS). The RSs are allocated during the ISSUE phase, denoted as RAT (Register Alias Allocation Table) in INTEL microarchitectures, as follows: an instruction if fetched from the QUEUE of decoded instructions and ISSUED if there is a free entry in the ROB ( head and tail of the ROB queue do not match); the instruction is moved into a RS (if available) when all of its operands are available. Dispatched instrutions are move to the UF – LOAD/STORE buffers one at a time.

**States** are labelled as follows:

WO: Waiting for Operands (at least one of the operands is not available)

RE: Ready for Execution (all operands are available)

DI: Dispatched (posted to a free RS)

EX: Execution (moved to a load/store buffer or to a matching and free UF)

WB: Write Back (result is ready and is returned to the Rob by using in exclusive mode the Common Data Bus CDB)

RR: Ready to Retire (result available or STORE has completed)

CO: Commit (result is copied to the final ISA register)

**State transitions** happen at the following events:

 *from* QUEUE *to* WO: ROB entry available, operand missing

*from* QUEUE *to* RE: ROB entry available, all operands available

*loop at* WO: waiting for operand(s)

*from* WO *to* RE: all operands available

*loop at* RE: waiting for a free RS

*from* RE *to* DI: RS/LOAD-STORE Buffer available

*loop on* DI: waiting for a free UF/waiting for cache (if not pilined)

*from* DI *to* EX: UF available – cache available

*loop at* EX: multi-cycle execution in a UF, or waiing for cache, or waiting for CDB

*from* EX *to* WB: result written to the ROB with exclusive use of CDB

*from* EX *to* RR: STORE completed, branch evaluted

*loop at* RR: instruction completed, not at the head of the ROB

*from* RR *to* CO: instruction at the head of the ROB, no exception raised

**Resources***Register-to-Register* instructions hold resources as follows:

ROB: from state WO (or RE) up to CO, inclusive;

RS: state DI

UF: EX and WB

*Load/Store* instructions hold resources as follows:

ROB: from state WO (or RE) up to CO, inclusive;

Load buffer: from state DI up to WB

Store buffer: from state DI up to EX (do not use WB)

**Forwarding**: a write on the CDB (WB) makes the operand available to the consumer in the same clock cycle. If the consumer is doing a state transition from QUEUE to WO or RE, that operand is made available; if the consumer is in WO, it goes to RE in the same clock cycle of WB for the producer.

**Branches**: they compute Next-PC and the branch condition in EX and optionally forward Next-PC to the “in-order” section of the pipeline (Fetch states) in the next clock cycle. They do not enter WB and go to RR instead.