COM522T : Computer Architecture

Tomasulo Algorithm Simulation

Project Report

Done by:
Firoz Mohammad CED17I017
Vaibhav Singhal CED17I040

INTRODUCTION:

Tomasulo's algorithm is a computer architecture hardware algorithm for dynamic scheduling of instructions that allows out-of-order execution and enables more efficient use of multiple execution units.

The major innovations of Tomasulo's algorithm include register renaming in hardware, reservation stations for all execution units, and a common data bus (CDB) on which computed values broadcast to all reservation stations that may need them. These developments allow for improved parallel execution of instructions.

Instruction life cycle

The three stages listed below are the stages through which each instruction passes from the time it is issued to the time its execution is complete.

Stage 1: issue

In the issue stage, instructions are issued for execution if all operands and reservation stations are ready or else they are stalled. Registers are renamed in this step, eliminating WAR and WAW hazards.

- Retrieve the next instruction from the head of the instruction queue. If the instruction operands are currently in the registers, then
    - If a matching functional unit is available, issue the instruction.
    - Else, as there is no available functional unit, stall the instruction until a station or buffer is free.
- Otherwise, we can assume the operands are not in the registers, and so use virtual values. The functional unit must calculate the real value to keep track of the functional units that produce the operand.

Stage 2: execute

In the execute stage, the instruction operations are carried out. Instructions are delayed in this step until all of their operands are available, eliminating RAW hazards.

- If one or more of the operands is not yet available then: wait for operand to become available on the CDB.
- When all operands are available, then: if the instruction is a load or store
    - Place it in the load/store buffer
        - If the instruction is a load then: execute as soon as the memory unit is available
        - Else, if the instruction is a store then: wait for the value to be stored before sending it to the memory unit
- Else, the instruction is an arithmetic logic unit (ALU) operation then: execute the instruction at the corresponding functional unit

Stage 3: write result

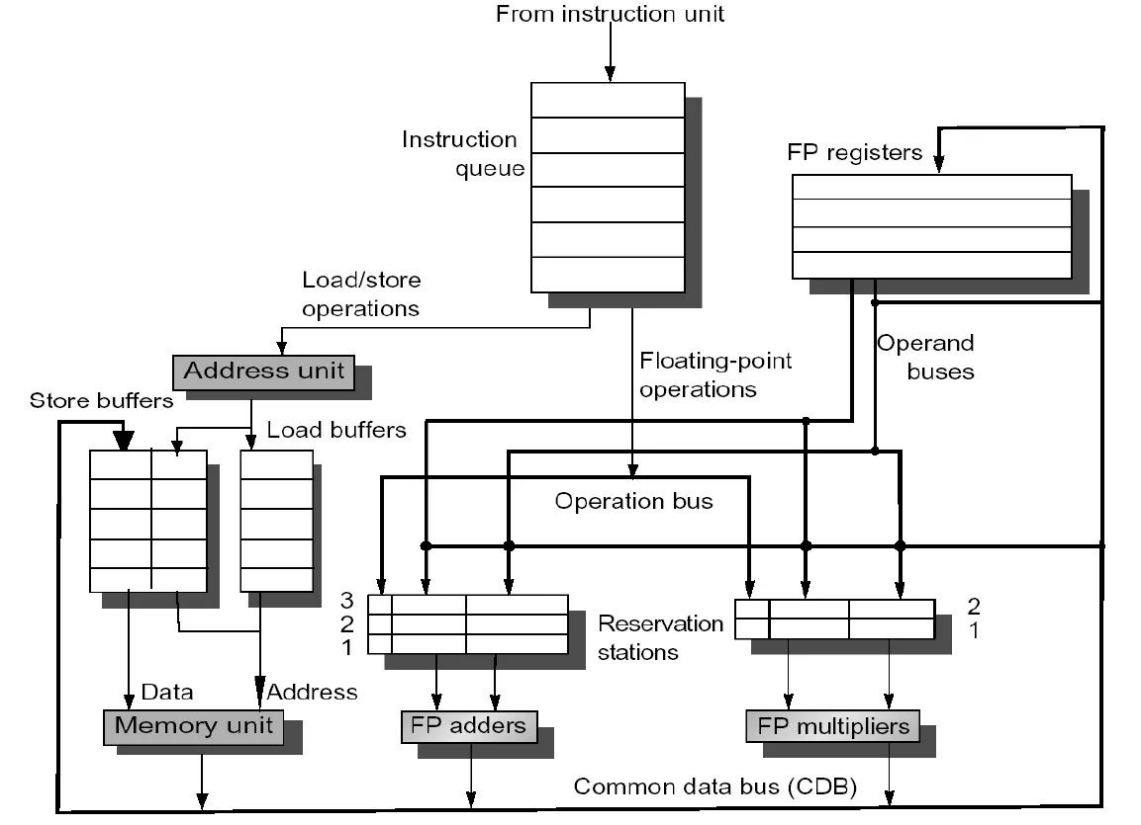
In the write Result stage, ALU operations results are written back to registers and store operations are written back to memory.

- If the instruction was an ALU operation
    - If the result is available, then: write it on the CDB and from there into the registers and any reservation stations waiting for this result
- Else, if the instruction was a store then: write the data to memory during this step

Instruction Set Architecture (ISA) :

OPCODE	Description	Usage	Explanation	Machine Cycle
0000	Addition	ADD Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 + Rsrc1	F D E(1-6) = 8
0001	Addition with Carry	ADC Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 + Rsrc1 + CF	F D E(1-6) = 8
0010	Subtraction	SUB Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 - Rsrc1	F D E(1-6) = 8
0011	Subtraction With Barrow	SBB Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 - Rsrc1 -1	F D E(1-6) = 8
0100	Multiplication	MUL Rdst, Rsrc2, Rsrc1	{R15, Rdst} = Rsrc2 * Rsrc1	F D E(1-11) = 13
0101	Floating Point Addition	FADD Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 + Rsrc1	F D E(1-21) = 23
0110	Floating Point Subtraction	FSUB Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 - Rsrc1	F D E(1-21) = 23
0111	Floating Point Multiplication	FMUL Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 * Rsrc1	F D E(1-24) = 26
1000	Halt	HLT	Halt	F D E(1-4) = 6
1001	Register Complement	CMP Rdst, Rsrc1	Rdst = ~Rsrc1	F D E = 3
1010	Logical Bit Wise XOR	XOR Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 ^ Rsrc1	F D E = 3
1011	Logical Bit Wise NAND	NAND Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 ~& Rsrc1	F D E = 3
1100	Shift Right	SHR Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 >> Rsrc1 If Rsrc1 = 2 then shift the Rsrc2 in 2bit towards right	F D E(1-4) = 6
1101	Shift Left	LHR Rdst, Rsrc2, Rsrc1	Rdst = Rsrc2 << Rsrc1 If Rsrc1 = 2 then shift the Rsrc2 in 2bit towards Left	F D E(1-4) = 6
1110	Store	STR Rdst, 8-bit Address	[8-bit Address] = Rdst	F D E(1-2) = 4
1111	Load	LDR Rdst, 8-bit Address	Rdst = [8-bit Address]	F D E(1-2) = 4

Diagram :



Assumptions

1. **No reorder buffer:** No exceptions such as divide by zero, wrong branch prediction
2. **More than 1 dispatch:** Instruction selected at random
3. **Used only int reg for int operations and float reg for float operations**
4. **8-bit binary address is represented in decimal**
5. **Instructions do not contains (,)**
6. **1 clock cycle each for issue and write result**
7. **Memory refers to cache memory and it is assumed that it will always be a cache hit**
8. **Instructions are already present in the instruction queue**
9. **There is no address unit**
10. **The carry bit is assumed to be set**
11. **Number of reservation stations and their assumed tags have been listed below:**

RESERVATION STATION	NUMBER
ADD/SUB (INT)	3
MULT(INT)	3
ADD/SUB (F)	3
MUL(F)	3
L-R SHIFT	3
COMP	2
XOR	2
NAND	2
LOAD STORE	4

INDEX/RS_TAG/LDST_TAG	Reservation station/LD ST Buffer
1	ADD/SUB(INT)
2	ADD/SUB(INT)
3	ADD/SUB(INT)
4	MUL(INT)
5	MUL(INT)
6	MUL(INT)
7	ADD/SUB(F)
8	ADD/SUB(F)
9	ADD/SUB(F)
10	MUL(F)
11	MUL(F)
12	MUL(F)
13	L-R SHIFT
14	L-R SHIFT
15	L-R SHIFT
16	COMP
17	COMP
18	NAND
19	NAND
20	XOR
21	XOR
22	LD ST BUFFER
23	LD ST BUFFER
24	LD ST BUFFER
25	LD ST BUFFER

12. **More than 1 broadcast:** Slower one executed first (Priority table listed below)

PRIORITY	FU
0	LOAD STORE
1	FLOAT MUL
2	FLOAT ADD
3	INT MUL
4	INT ADD
5	SHIFT
6	COMP
7	XOR
8	NAND

Components

1. **INT Register :** Register to hold integer values
    a. **Busy :** Indicates whether the register is busy i.e., new value is to be written in it
        i. 0 : not busy
        ii. 1 : busy
    b. **Tag :** Indicates the reservation station/load store buffer from which it will receive the new value
        i. -1 : default sentinel value
        ii. 0 : Data is present in the data field is correct and the register is not waiting for any reservation station or load store buffer
        iii. 1-25 : Indicates the reservation station number(tag)/load store buffer number(tag) from which new data is to be obtained
    c. **Data :** To hold integer value
2. **FP Register :** Register to hold floating values
    a. **Busy :** Indicates whether the register is busy i.e., new value is to be written in it
        i. 0 : not busy
        ii. 1 : busy

b. **Tag :** Indicates the reservation station/load store buffer from which it will receive the new value
    i. -1 : default sentinel value
    ii. 0 : Data is present in the data field is correct and the register is not waiting for any reservation station or load store buffer
    iii. 1-25 : Indicates the reservation station number(tag)/load store buffer number(tag) from which new data is to be obtained
c. **Data :** To hold floating value

3. **Reservation Stations :**
    a. **RS Tag :** Indicates the tag of the reservation station
    b. **Busy :** Indicates whether the reservation station is busy
        i. 0 : not busy
        ii. 1 : busy
    c. **Opcode :** Indicates the operation to be performed
    d. **Tag1 :** Indicates the reservation station/load store buffer from which it will receive the new value of the operand 1 incase the value is yet to be calculated and updated in its respective field
        i. -1 : default sentinel value
        ii. 0 : Data is present in the value field is correct and is not waiting for any reservation station or load store buffer
        iii. 1-25 : Indicates the reservation station number(tag)/load store buffer number(tag) from which new data is to be obtained
    e. **Value1 :** Stores the value of operand 1 when available
        i. inf : default sentinel value
    f. **Tag2 :** Indicates the reservation station/load store buffer from which it will receive the new value of the operand 2 incase the value is yet to be calculated and updated in its respective field
        i. -1 : default sentinel value
        ii. 0 :Data is present in the value field is correct and is not waiting for any reservation station or load store buffer
        iii. 1-25 : Indicates the reservation station number(tag)/load store buffer number(tag) from which new data is to be obtained
    g. **Value2 :** Stores the value of operand 2 when available

i. inf : default sentinel value

h. **Issued :** Refers to the dispatching of the instruction into the fetch unit of execution stage
   i. 0 : Not dispatched
   ii. 1 : Dispatched

4. **Load/Store Buffer :**
   a. **Ldst_tag :** Indicates the tag of load/store buffer
   b. **Busy :** Indicates whether the buffer is busy
      i. 0 : not busy
      ii. 1 : busy
   c. **Opcode :** Indicates the operation to be performed
   d. **Address :** Holds memory address to be read or write at that location
   e. **Tag :** Indicates the reservation station/load store buffer from which it will receive the new value that is to be stored in case of store instruction
      i. -1 : default sentinel value
      ii. 0 : Value is present in the value field is correct and not waiting for any reservation station or load store buffer
      iii. 1-25 : Indicates the reservation station number(tag)/load store buffer number(tag) from which new data is to be obtained
   f. **Value :** Holds value to be stored in memory
   g. **Issued :** Refers to the dispatching of the instruction into the fetch unit of execution stage
      i. 0 : Not dispatched
      ii. 1 : Dispatched
5. **Result Buffer :** Part of the last stage of the FDE of functional unit which stores the result before writing it into the Common data bus in the write result stage
   a. **Busy :** Indicates whether the result buffer is busy
      i. 0 : not busy
      ii. 1 : busy
   b. **Tag :** Stores the rs/buffer tag in which the output must be written
      i. -1 : default sentinel value
   c. **Result :** Stores the final result
      i. inf : default sentinel value

   d. **Address :** Storing the address in case of a store operation
      i. -1 : default sentinel value
6. **FU Pipeline stages :** Indicates the pipelined fu stages, the number denotes the current cycle count of an instruction inside a particular pipeline stage
7. **CDB :** A common data bus at which results are written from result buffer in priority order
   a. **Busy :** Indicates whether the common data bus is busy
      i. 0 : not busy
      ii. 1 : busy
   b. **Tag :** stores tag of reservation stations/load store buffer/INT register/FLT register which is to be updated
   c. **Result :** Store results of instruction