**PREM KRISHNA CHETTRI**

**Computer Architecture Assignment 4 Submission Date: 29th Oct ‘15**

**Solution 1.1**. Primarily two factors limit the number of instructions that can be dispatched.

1>Number of functional units available for the execution of an instruction.

2> Amount of parallelism in the instructions.

**Solution 1.2.** Instruction has to be finally available at architectural register for the ISA to determine the instruction value for further execution. Moreover, we have a limited number of physical register and we needed to free these physical register as soon as it finishes the execution for the other instruction to use it.

**Solution 1.3.** Register Alias Table, Rename table, Status bit

**Solution 1.4.** Forwarding is basically forwarding data from one functional unit to all other who are waiting for the results within the same clock cycle, where as completion is the basically the forwarded result written to reorder buffer.

If the forwarded result satisfies all the dependencies then the value in the re-order buffer is changed as accordingly and leads to the completion.

All dependencies are resolved and there is no instruction waiting for the value from the other instruction.

**Solution 1.5.** We may keep some form of thread Id associated with each of the instruction so that when the instruction retires it will able to tell where from it has been issued.

**Solution 1.6.** This is false dependency. Using register renaming, we can completely avoid any dependency over its registers. We assign two different physical register for source and destination R3 while register renaming.

**Solution 2.1.**

All Architecture register is being renamed to physical register and hence renamed code fragment will be.

I1:  MOVI P6, #4000

I2:  MOV P7, P6

I3:  LOAD P8, P7, #9

I4:  SUB P9, P7, P8 // P7, P8 Can now be freed

I5:  MUL P7, P9, P6 // P6 Can be freed now

I6:  STORE P6, P9, #11

I7:  LOAD P6, P9, #33 // P9 Can be freed now

I8:  AND P8, P6, P7

Grantt Chart CYCLES

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |  |
| F | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ISQ |  |  | 1 | 2 | 32 | 43 | 543 | 654 | 7654 | 8 7654 | 87654 | 8765 | 8765 | 87 | 8 | 8 | 8 | 8 | 8 | 8 |  |  |  |  |  |  |  |  |  |
| ALU |  |  |  | 1 |  | 2 |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  | 8 |  |  |  |  |  |  |  |  |
| LS0 |  |  |  |  |  |  |  | 3 |  |  |  |  |  | 6 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LS1 |  |  |  |  |  |  |  |  | 3 |  |  |  |  |  | 6 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LS2 |  |  |  |  |  |  |  |  |  | 3 |  |  |  |  |  | 6 |  |  | 7 |  |  |  |  |  |  |  |  |  |  |
| M0 |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |
| RET |  |  |  |  | 1 |  | 2 |  |  |  | 3 |  | 4 |  |  |  |  | 5 | 6 | 7 |  | 8 |  |  |  |  |  |  |  |

RAT

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |  |  |  |
| R0 | P0 | 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R1 | P1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R2 | P2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R3 | P3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R4 | P4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R5 | P5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Allocated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Renamed

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Solution 3.1.**

2 Operand ISA

MOV R1, R2

SUB R1, R3

MUL R2, R1

LDI R1, #100

ADD R5, R4

MOV R6, R5

AND R6, R4

XOR R5, R6

LDI R6, #100

**Solution 3.2.**

**Renamed 3-Operand Instructions**

SUB P32, P2, P3

MUL P33, P32, P2

LDI P32, #100 // Release P32 after this

ADD P32, P5, P33

ADD P34, P32, P33 // Release P33 after this

XOR P33, P34, P32

LDI P34, #100

**Solution 3.3.**

**Renamed 2-operand Instructions.**

MOV P32, P2

SUB P32, P3

MUL P2, P32

LDI P32, #100 // Release P32 After this

ADD P5, P4

MOV P32, P5

AND P32, P4

XOR P5, P32

LDI P32, #100

**Solution 3.4.**

**Solution 3.5.** As from the paper, they found some benchmark, which explains that, for zeroing a register; add operation is slower then subtracting the register content. However, the best way to zeroing the register is by x-or to itself.

//x-or a register to itself has some problem. If there is a data dependency, it have to wait till all the dependencies resolves before it goes for the operation.

By the usage of the register renaming, we can get two physical register with same value and we can x-or them to produce the zero valued register. The paper further explains the implementation in Sandy Bridge processor, where they have integrated a mechanism in renaming register to detect the condition that always zeros the register and rather then processing through the execution steps, it can directly zero the register itself.