Superscalar Processor

Design Report

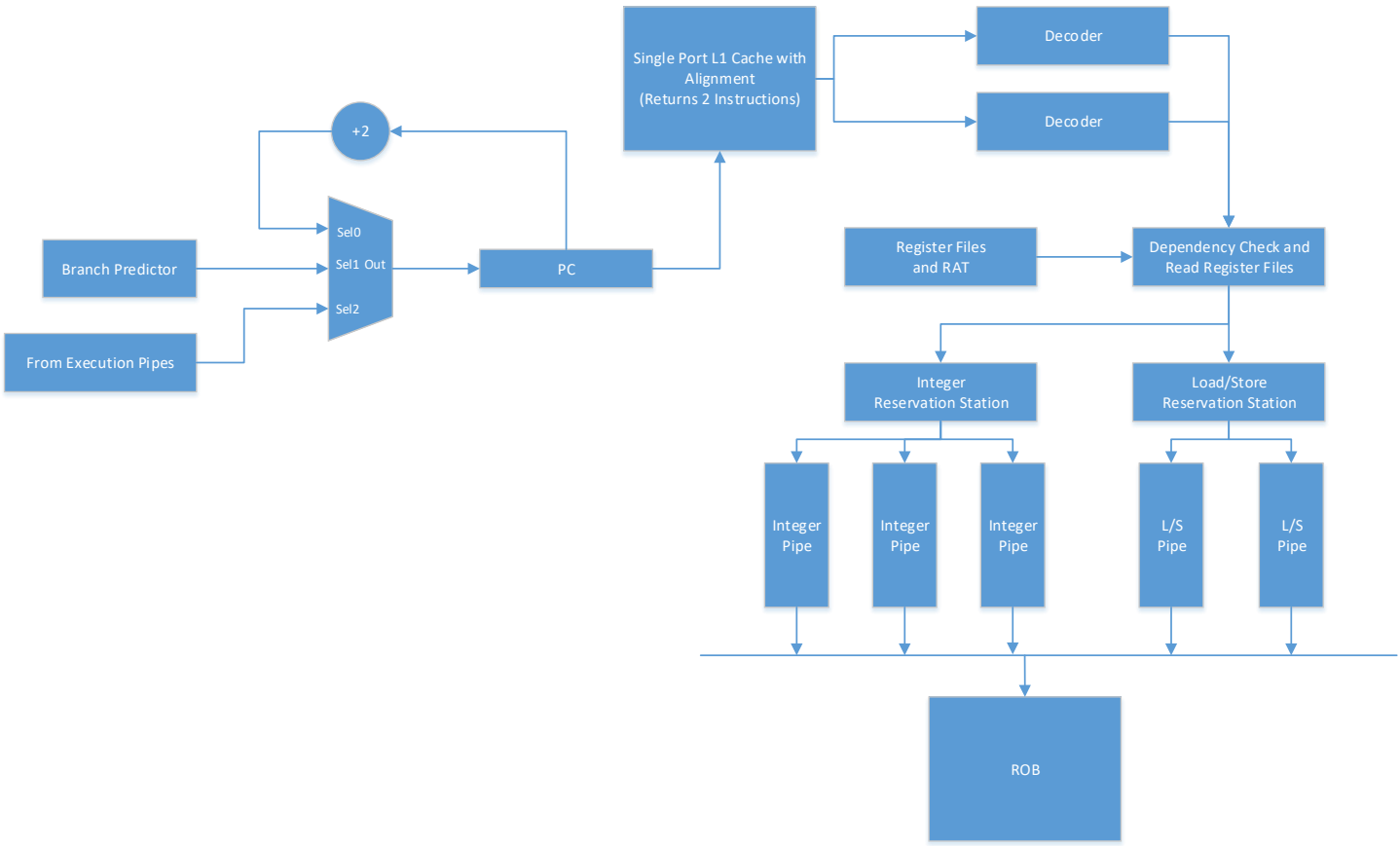
Yogesh Mahajan
14D070022

Sudipto Banerjee
14D070028

Balraj Parmar
14D070001

Eswar Bawineni

Design Overview



Physical Register Size	24
Renamed Flag Register File	24
ROB Size	32
Integer Reservation Stations	16
Load/Store Reservation Station	8

Fetch Stage

Instruction memory returns two consecutive instructions starting from provided PC. Fetched Instructions are checked with BTA and Branch History Table. If branch is predicted to be taken and address is available in BTA then PC is updated accordingly. If address is not available for always taken branches then fetcher will be stalled because there is no use in fetching unnecessary instructions. All the instructions after branch are marked with speculative bit in ROB to avoid completion before resolving branch. Fetcher will be stalled for decoding of

Output of fetch stage is stored in register

Valid Bit	Instruction1	Valid Bit	Instruction2

Special Case

First instruction is branch and predictors predicts it to be taken/is always taken then second instruction is useless. In this case second instruction in packet is marked with invalid and PC is updated in next cycle.

Validity is checked using computed target address combining both the instructions. We can add special checking field for such instructions if these are very rare, mostly at the end of nested loops

Decode

Stage One

Determine instruction operands and destination along with destination address and type of instruction. If only the first instruction is branch and it speculated to be taken then discard second instruction. If both are branches then raise multitargeted bit for both so that their combined target address can be verified later using above mentioned method.

Valid	Multi Target	ALU instructions	R1	OP1	OP2	Valid	Multi Target	ALU instructions	R2	OP3	OP4

Stage Two

Stage two performs intra dependency check and assign registers/ data

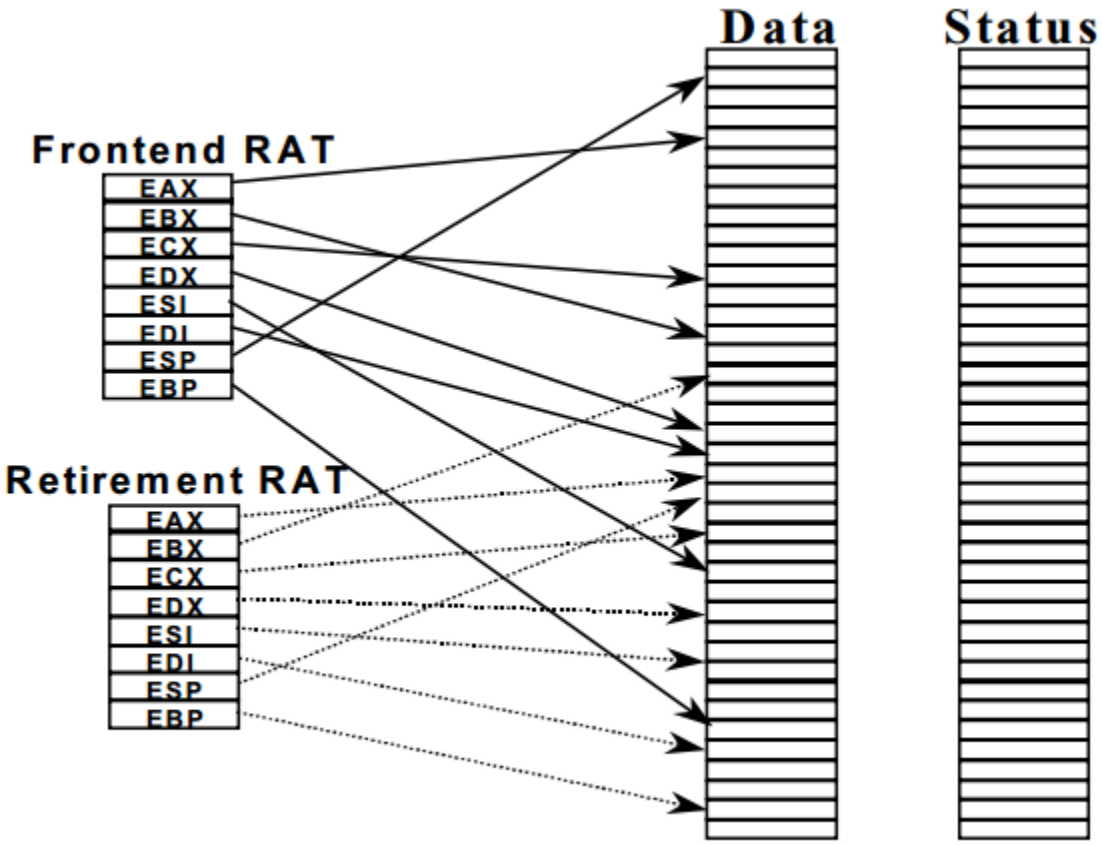
```
// Decode Stage Two
    // For instruction 1
    Get pointers of operand registers from frontend RAT
    Copy data form register file to output field using pointers

// for instruction 2
    //  for both operands
    if(OP == R1)
        Copy R1's pointer for output filed of OP3 and clear data bit
    else
        Get data using pointer from RAT

// Register Reservation for Destination
    Get pointer of free register from queue and make corresponding entry to RAT
    Repeat same procedure for renamed flag registers
```

Register Renaming

There are to RATs frontend and retirement RAT. Frontend RAT is used for issuing instructions and retirement RAT holds the value indices to the registers which are visible to programmer. Renamed flag registers don't need RATs as they are sequentially assigned to each instruction.



Register renaming is similar to Intel's NetBurst architecture used in Pentium 4

Reservation Station

Reservation Station for Integer Instructions

Valid	Speculative	ALU Instructions	Destination	destination Pointer	Write	Operand one	OP1 data/pointer

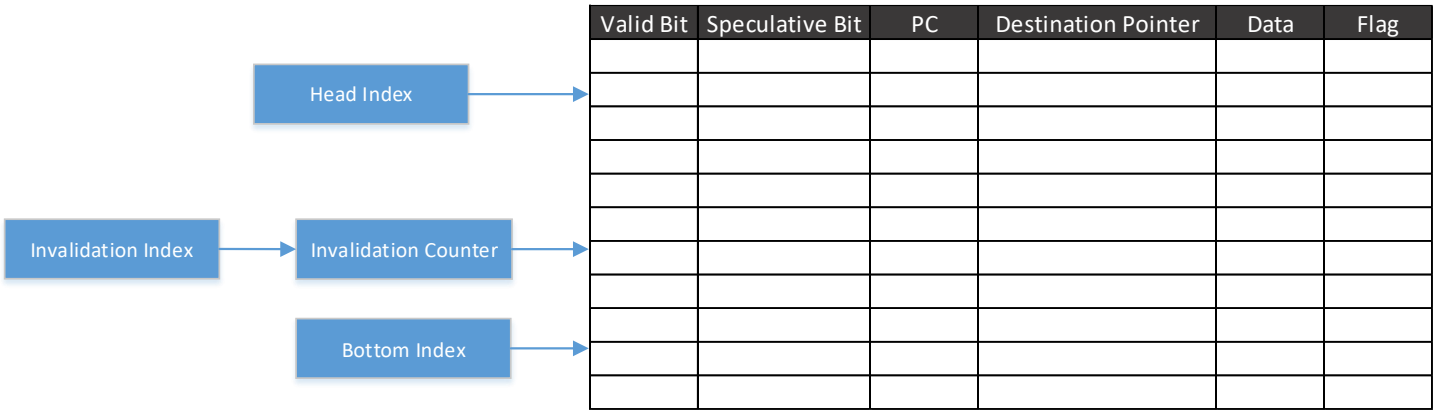
Operand 2	OP2 data/pointer	Flag/pointer	Flags	Branch Direction	Branch Target	Multitarget Branch	

Reservation Station for Load/Store Instructions

Valid	Speculative	Instruction	Destination	Destination Pointer	Write

Operand one	OP1 data/pointer	Operand 2	OP2 data/pointer	Flag/pointer	Flags

ROB



Removing wrongly speculated instructions from the ROB

Whenever any branch or load is confirmed to be wrongly speculated then all the entries above that branch in ROB are to be marked invalid. Current head of ROB +2 (for instructions in fetch stage who will reserve entries in next cycle) is copied to the counter which will make two instructions invalid in each stage freeing up the space cycle by cycle.