Microprocessor Microarchitecture
Dependency and OOO Execution

Lynn Choi

Dept. Of Computer and Electronics Engineering

Three Forms of Dependence

- ***True dependence (Read-After-Write)***
  - Also called flow dependence
  - Require pipeline interlock
  - Data bypass (forwarding) can reduce the producer latency
    - Make values generated by FUs immediately available
- ***Output dependence (Write-After-Write)***
- ***Anti dependence (Write-After-Read)***
  - Both of them are called false dependencies
  - Require pipeline interlock or register renaming

In-Order Pipeline

- ***In-order issue***
  - If an instruction is stalled in the pipeline, following instructions cannot proceed. However, once issued to FUs, in general the instruction need not be stalled.
  - Instruction can complete out-of-order
- ***Dependency resolution mechanism***
  - Pipeline interlock
    - Need reg-id comparators between sources and destinations of instructions in REG stage and the destinations of instructions in the EXE and WRB stages
    - Comparators needed for both interlock and *bypass*
  - Scoreboard
    - A busy bit for each register
    - For long latency operations such as MEM operations
    - Instead of comparators, you need to check scoreboard for operand availability
    - Comparators are still needed for bypass!

Example

- ***FET-DEC-REG-EXE-WRB***
- ***What kind of dependence violations are possible?***
  - Single-issue 5-stage in-order pipeline with the following pipelined FUs
    - 2 INT unit (1 cycle INT operation)
    - 1 FP unit (4 cycle FP operation)
    - 2 MEM pipelines (2 cycle MEM operation)
- ***How many comparators do you need for the previous example?***
    - RAW
      - 2 srcs * 2 stages (E, W) * 2 INT = 8
      - 2 srcs * 5 stages (E1, E2, E3, E4, W) * 1 FP = 10
      - 2 srcs * 3 stages (E1, E2, W) * 2 MEM = 12
    - WAW
      - 1 dest * 3 stages (E1, E2, E3) * 1 FP = 3
      - 1 dest * 1 stages (E1) * 2 MEM = 2
  - WAW hazard can happen only for MEM and FP pipelines.

Out-Of-Order Machines

- ***Anti-dependence can happen in OOO machines***

```
DIV  F0, F2, F4
ADD  F10, F0, F8
SUB F8, F8, F14
```

- ***Different approaches***
  - Scoreboarding
  - Tomasulo’s Algorithm
  - Register Update Unit

Scoreboarding - CDC6600 -

- ***Scoreboard***
  - One bit per register indicates whether or not there is a pending update
- ***Pipeline stalls on WAW and WAR dependences***

- ***FET-DEC/ISS-REG-EXE-WRB***
  - DEC/ISSUE stage: check for WAW and structural hazards
    - (Centralized) instruction window between ISS and REG stages
    - Pipeline stalls on *output dependence* by checking scoreboard
      - Allows only 1 pending update
    - Pipeline also stalls if there is no empty entry in the instruction window
  - REG stage
    - Resolve RAW hazards
    - Instructions are sent to FUs out of order
  - WRB stage:
    - Once the execution completes, check for WAR hazards

Tomasulo's Algorithm - Reservation Station

- ***Used in IBM 360/91 floating point unit (1967)***

- ***Three ideas***
  - OOO execution using *reservation stations (RS)*
    - Distributed instruction windows
  - *Register renaming* to remove anti and output dependencies
    - Read available input operands from RF and store them into RS (WAR removal)
    - Assign new storage for output (WAW removal)
    - Pipeline does not stall on WAW and WAR hazards
  - Data forwarding using *common data bus*
    - Bypass the data directly to the waiting instructions in RS
    - Both register file and RS (source and dest) monitor the result bus and update data when a matching tag is found

Tomasulo's Algorithm

- ***FET-DEC/REN/ISS-REG-EXE-WRB-COM***
  - REN/ISS stage: check structural hazard (reservation station entry) and read available operands from register file (register renaming for WAR) and assign RS entry for destination (WAW hazard)
  - REG stage: monitor common data bus and read operands into RS if there is a match; determine highest priority operations among ready operations (wakeup)
  - EXE: execute and forward result to RS and RF

- ***Instruction buffers***
  - Instruction queue between FET and DEC/ISS stages
    - can be omitted
  - *Reservation station* between ISS and REG stages
  - *Reorder buffer* between WRB and COM stages
    - not in original proposal (IBM 360/91)

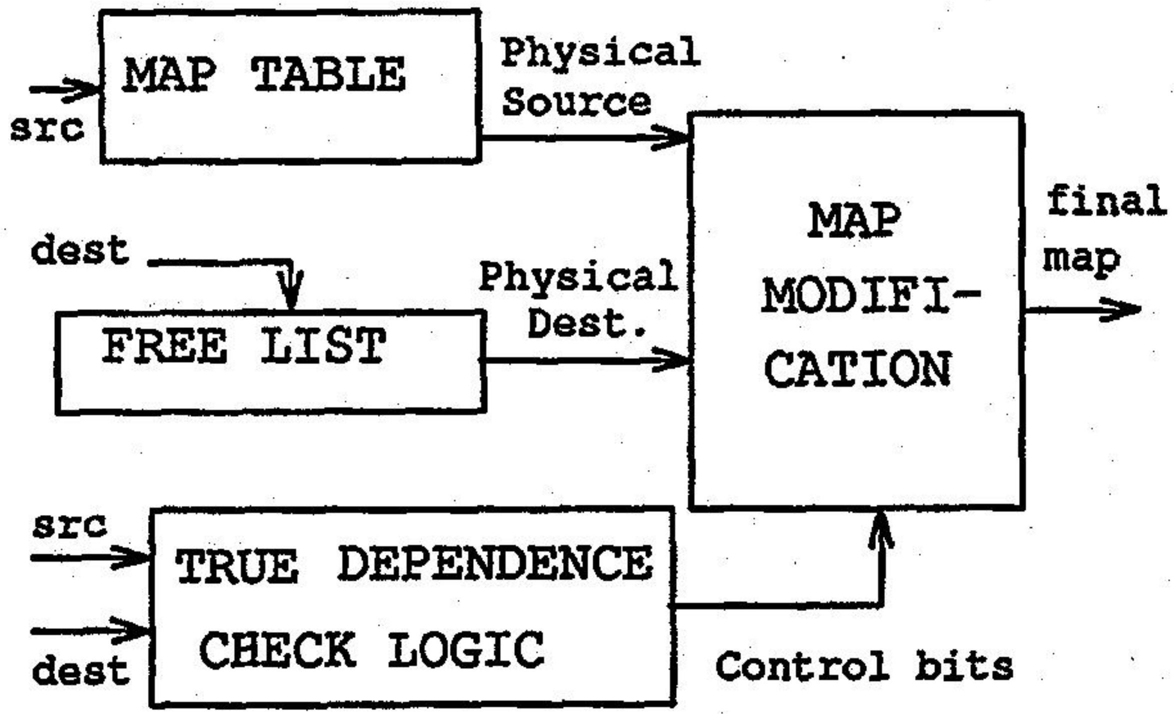
Renaming

- ***Removes anti and output dependencies***
  - Allows more than one pending update
- ***Several forms of renaming***
  - Tomasulo's algorithm
    - Reservation station for additional storage for name dependencies and common data bus for data bypass
  - Reorder buffer with associative lookup
    - Associative lookup maps the reg id to the reorder buffer entry as soon as an entry is allocated
  - Register map table with separate physical register file
    - Register map table (DEC 21264)
    - Register alias table (Intel P6)

Renaming

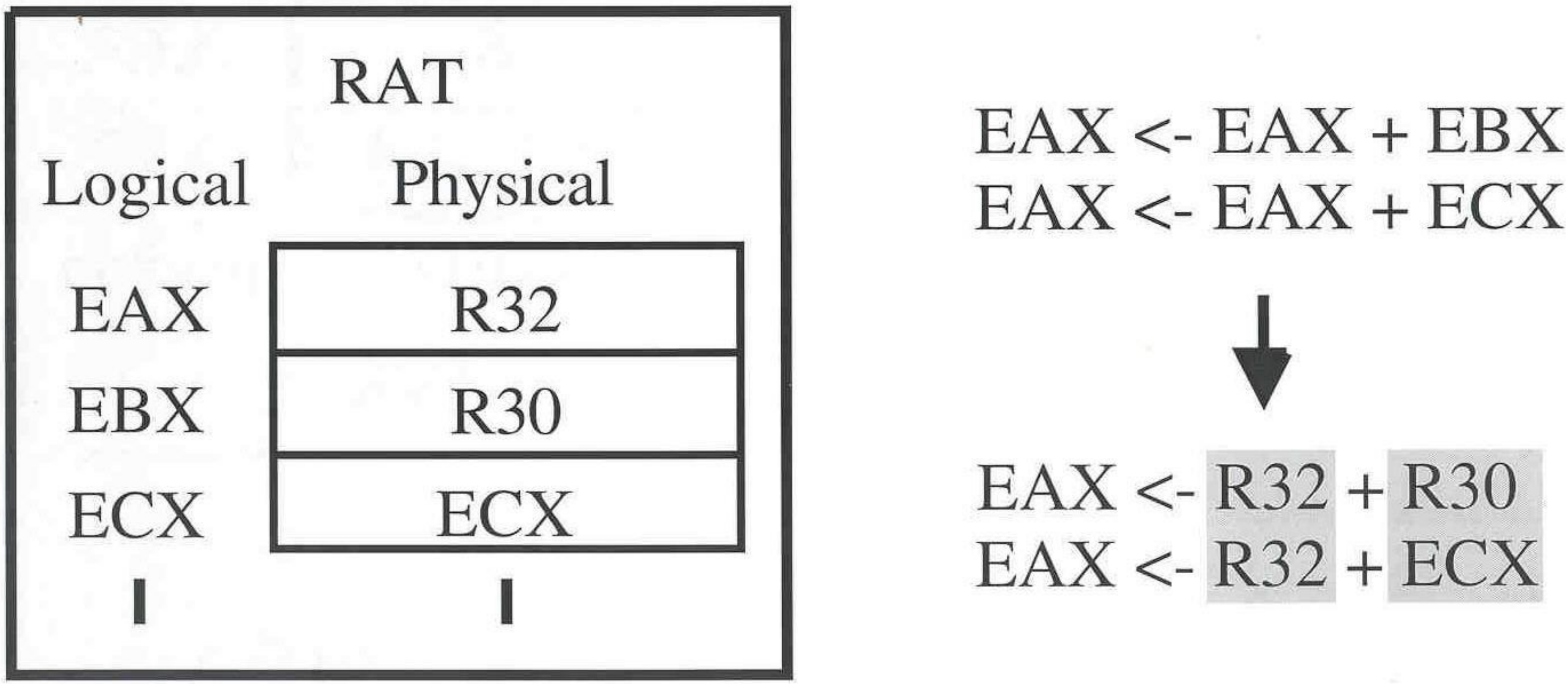
- ***Assign one physical register for every instruction with a destination register***
  - With 80 instructions in flight (reorder buffer size)
  - You need roughly 80 physical registers (except branch and stores)
  - physical registers are single-assignment registers
- ***Register renaming involves data dependence checking among the instructions that are simultaneously being renamed***
  - Renaming bandwidth limited by
    - Data dependence checking
    - Number of read ports needed for register map table

Renaming

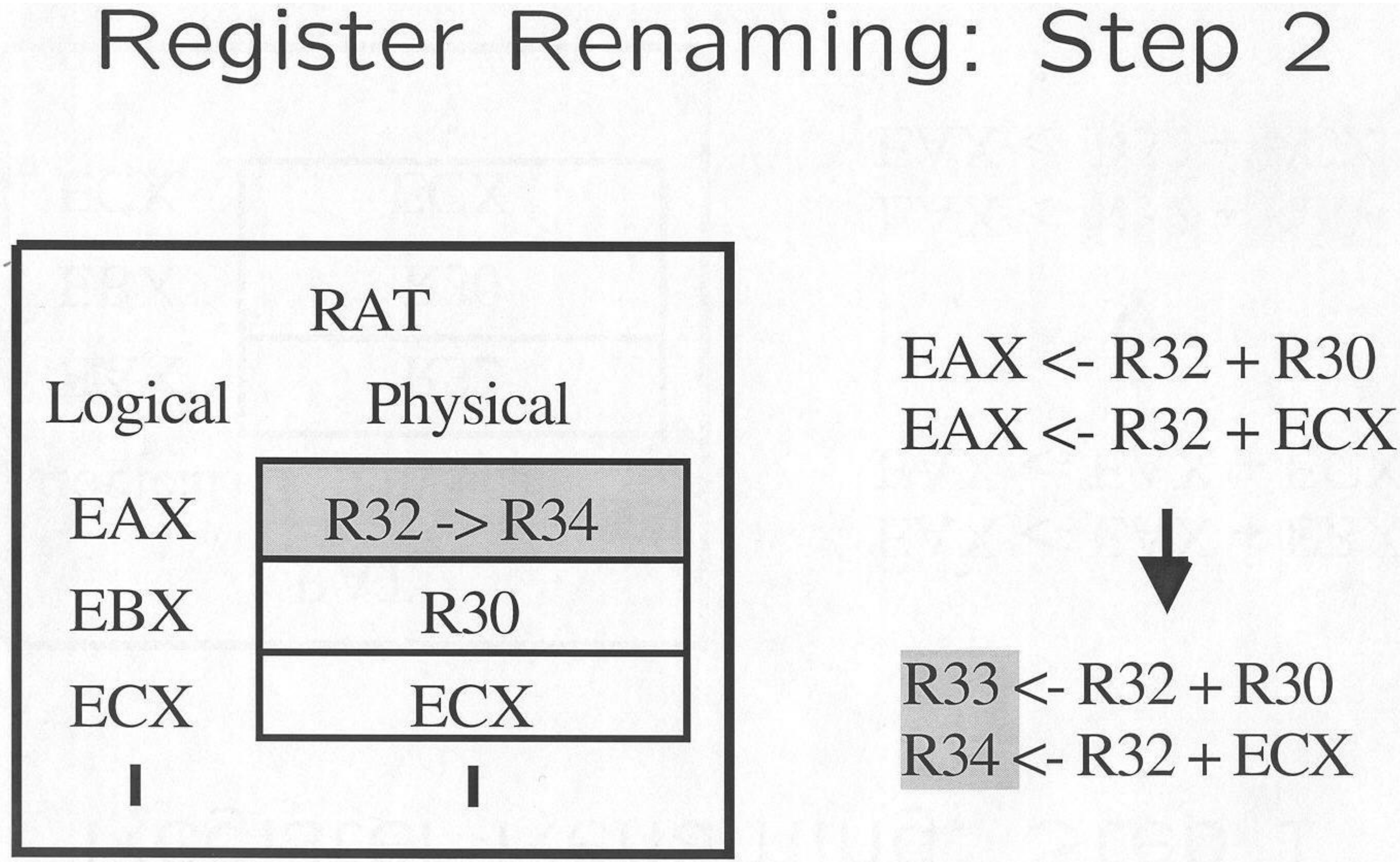

src
MAP TABLE
Physical
Source
dest
FREE LIST
Physical
Dest.
MAP
MODIFI-
CATION
final
map
src
dest
TRUE DEPENDENCE
CHECK LOGIC
Control bits

Rename Example (P6)

Register Renaming: Step 1



Rename Example (P6)



Rename Example (P6)

Register Renaming: Step 3

RAT

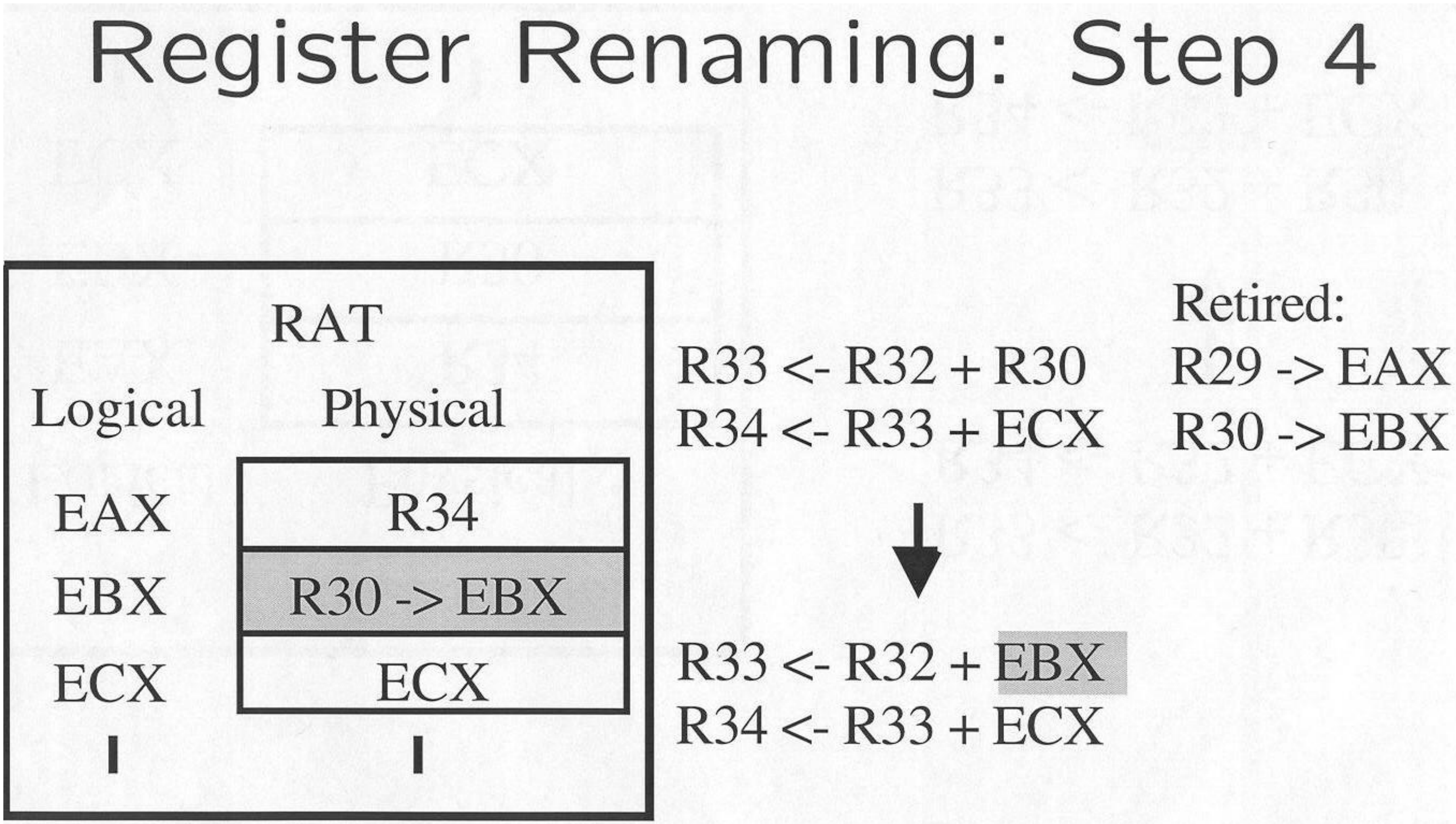
Logical	Physical
EAX	R34
EBX	R30
ECX	ECX

R33 <- R32 + R30
R34 <- R32 + ECX

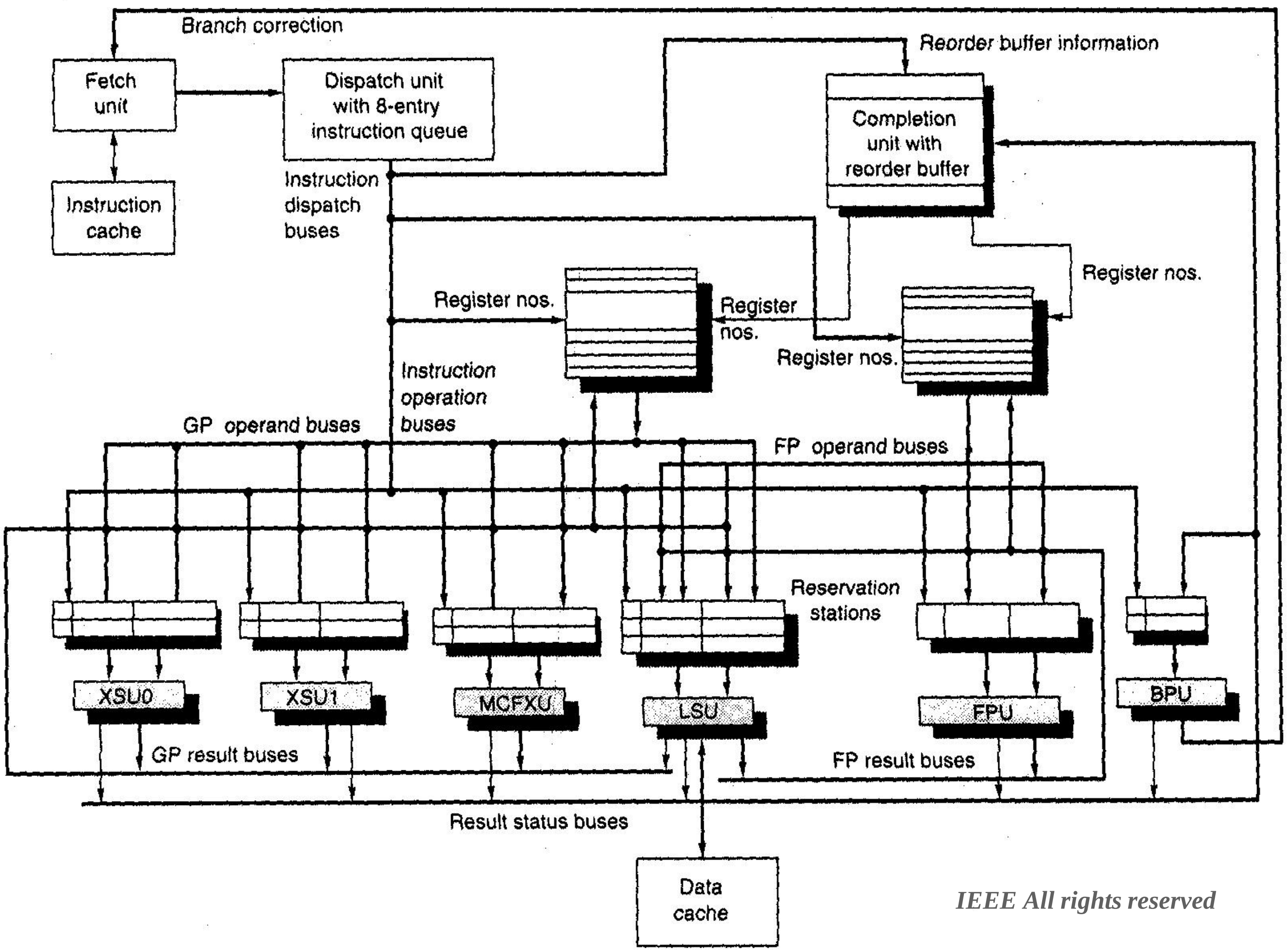
↓

R33 <- R32 + R30
R34 <- R33 + ECX

Rename Example (P6)



PowerPC 620 - OOO example -



DEC 21264 - OOO example -

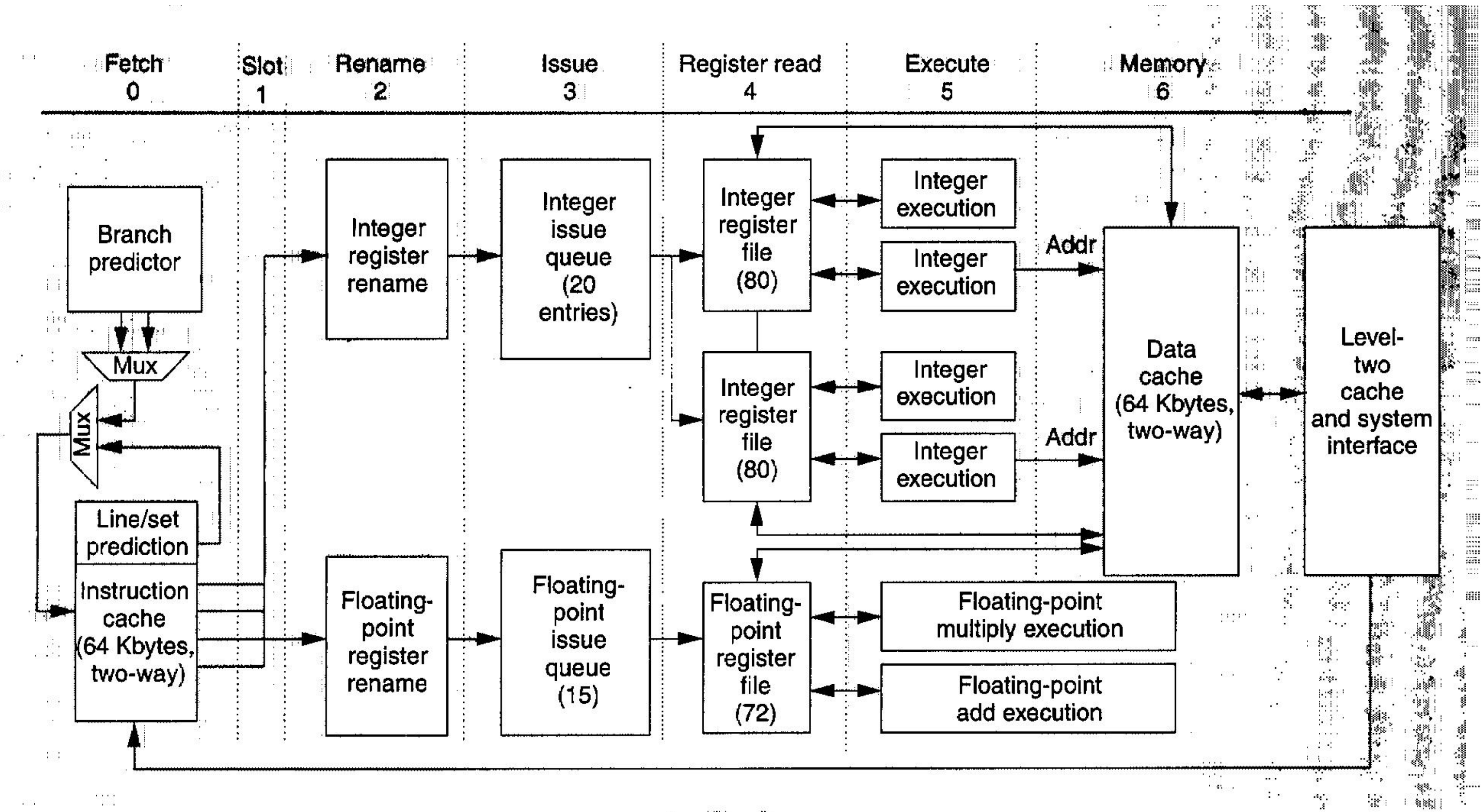
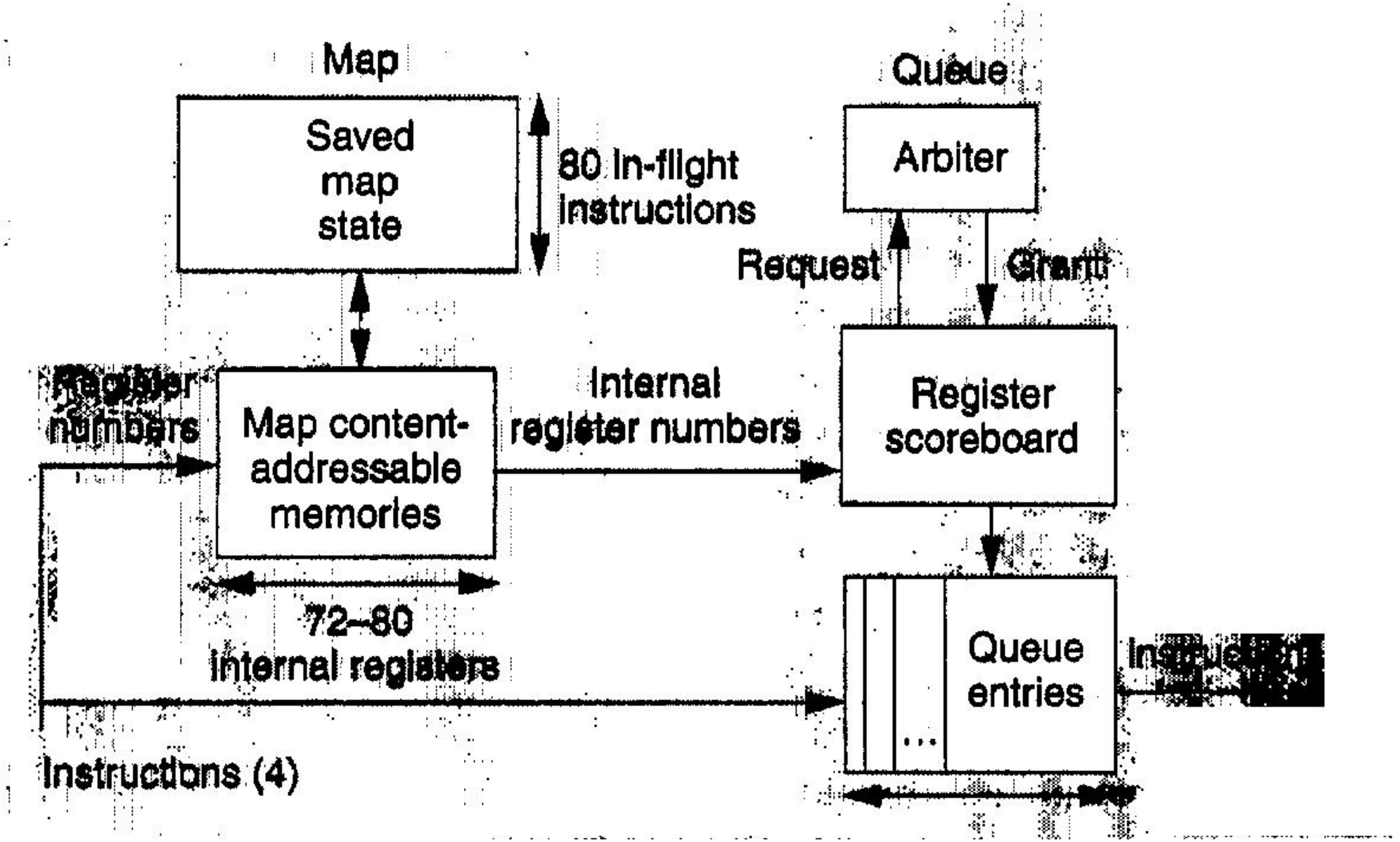
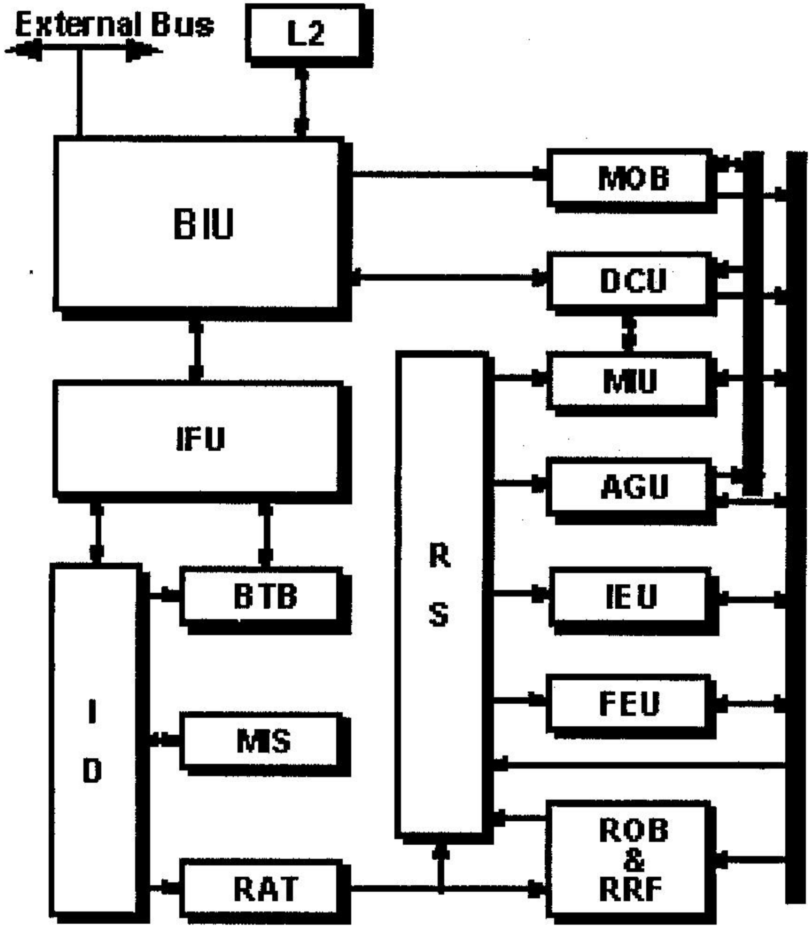


Figure 2. Stages of the Alpha 21264 instruction pipeline.

DEC 21264 - OOO example -



Intel P6 - OOO example -



- BIU: Bus Interface Unit
- IFU: Instruction Fetch Unit (includes ICach
- BTB: Branch Target Buffer
- ID: Instruction Decoder
- MIS: Microinstruction Sequencer
- RAT: Register Alias Table
- ROB: ReOrder Buffer
- RRF: Retirement Register File
- RS: Reservation Station
- IEU: Integer Execution Unit
- FEU: Floating point Execution Unit
- AGU: Address Generation Unit
- MIU: Memory Interface Unit
- DCU: Data Cache Unit (includes DCache)
- MOB: Memory ReOrder Buffer
- L2: Level 2 Cache

Exercises and Discussion

- ***There can be many instruction buffers in an OOO processor. Name those buffers and explain their functions.***
- ***What happens on a branch misprediction in OOO processors?***

Homework 2

- ***Read Chapter 3***
- ***Exercise***
  - 3.1
  - 3.2
  - 3.3
  - 3.4
  - 3.8
  - 3.13
  - 3.17