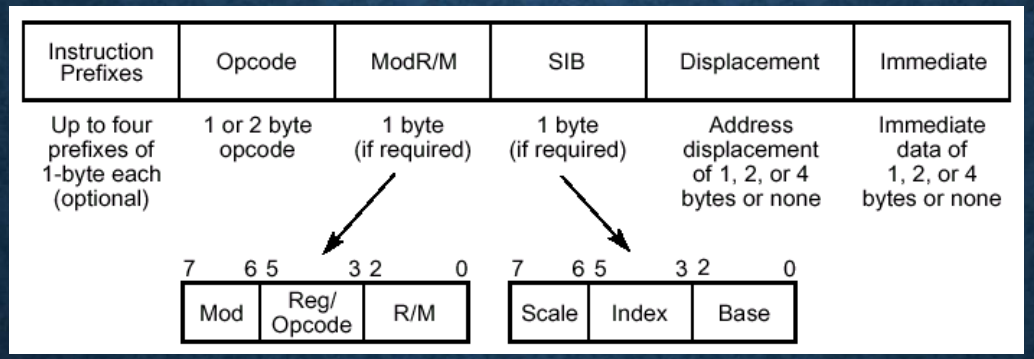
LECTURE 12

Out-of-order execution: Pentium Pro/II/III

EXECUTING IA32/IA64 INSTRUCTIONS FAST

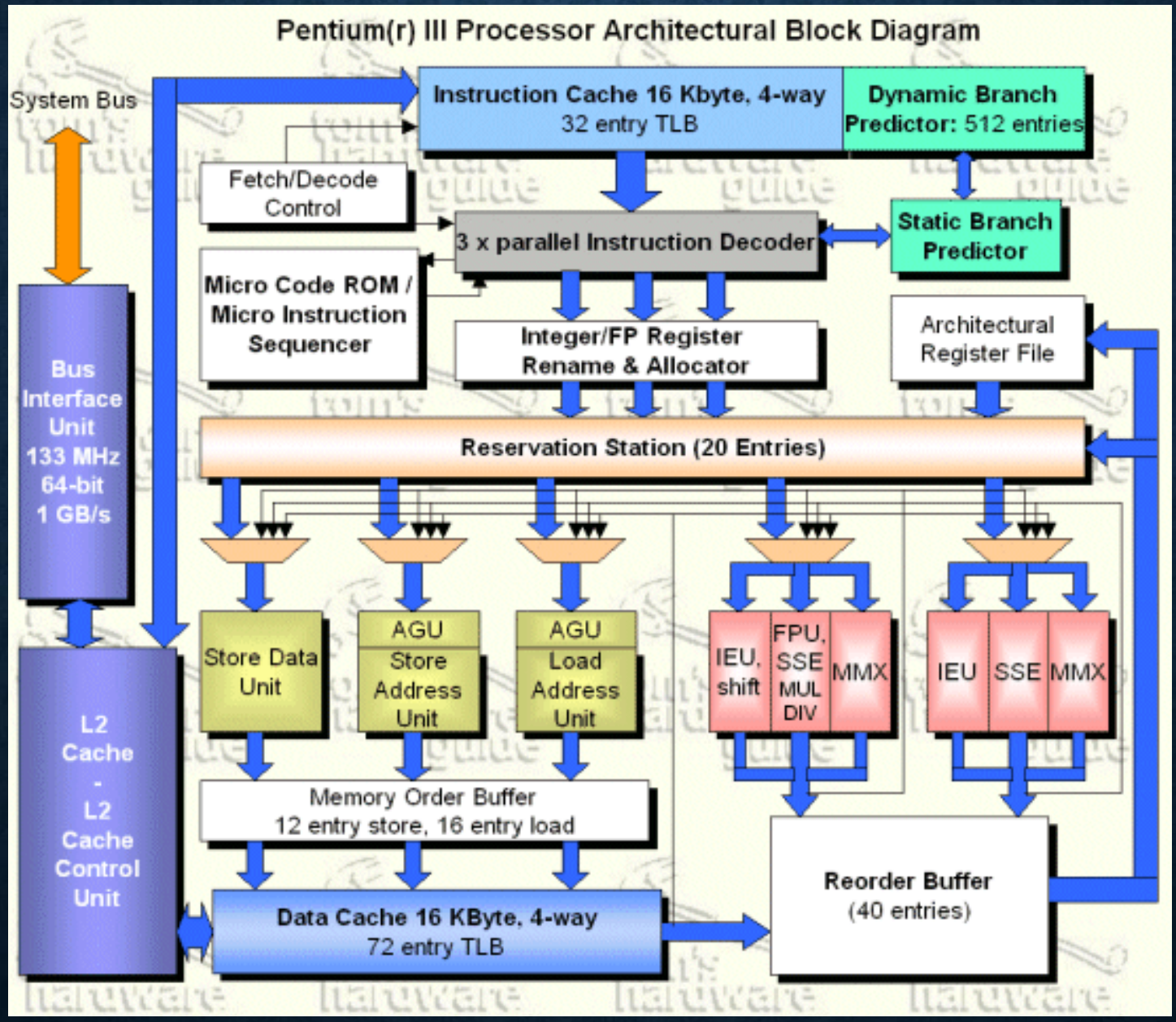
- Problem: Complex instruction set



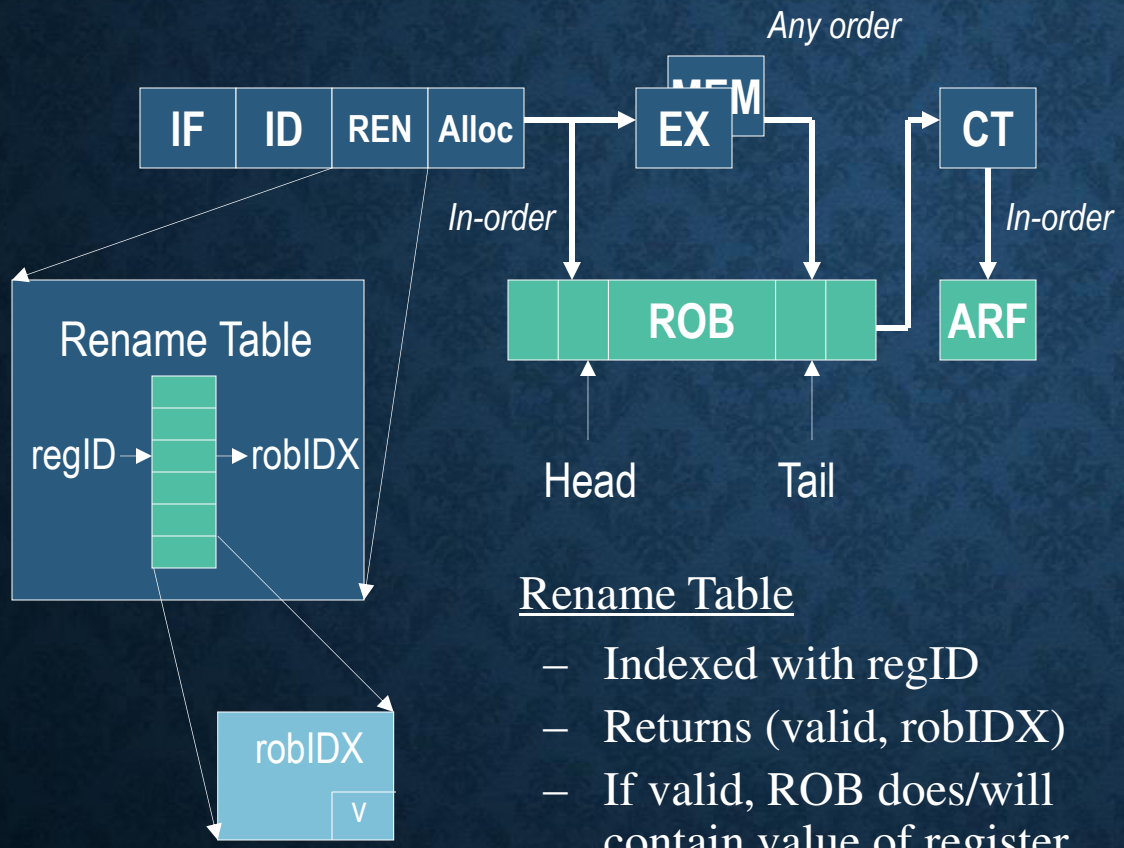
- Solution: Break instructions up into RISC-like micro operations
  - Lengthens decode stage; simplifies execute

PENTIUM PRO/II/III PROCESS STAGES

- The first stage consists of the instruction fetch, decode, convert into micro-ops, and reg rename
- The reorder buffer (ROB) is the buffer between the first and second stages
- The ROB is also the buffer between the second and third stages
- The third stage retires the micro-operations in original program order
  - Completed micro-operations wait in the reorder buffer until all of the preceding instructions have been retired


Pentium(r) III Processor Architectural Block Diagram
System Bus
Instruction Cache 16 Kbyte, 4-way
32 entry TLB
Dynamic Branch
Predictor: 512 entries
Fetch/Decode
Control
3 x parallel Instruction Decoder
Static Branch
Predictor
Micro Code ROM /
Micro Instruction
Sequencer
Integer/FP Register
Rename & Allocator
Architectural
Register File
Bus
Interface
Unit
133 MHz
64-bit
1 GB/s
Reservation Station (20 Entries)
Store Data
Unit
AGU
Store
Address
Unit
AGU
Load
Address
Unit
IEU,
shift
FPU,
SSE
MUL
DIV
MMX
IEU
SSE
MMX
L2
Cache
-
L2
Cache
Control
Unit
Memory Order Buffer
12 entry store, 16 entry load
Data Cache 16 KByte, 4-way
72 entry TLB
Reorder Buffer
(40 entries)

Pentium Pro pipeline overview



Rename Table

- Indexed with regID
- Returns (valid, robIDX)
- If valid, ROB does/will contain value of register
- If invalid, ARF holds value (no instruction in flight defines this register)

@ **Fetch (2 cycles)**

read instructions (16 bytes) from memory from IP (PC)

@ **Decode (3 cycles)**

Decode up to 3 instructions generating up to 6 μops

Decoder can handle 2 "simple" instructions and 1 "complex" instruction. (4-1-1)

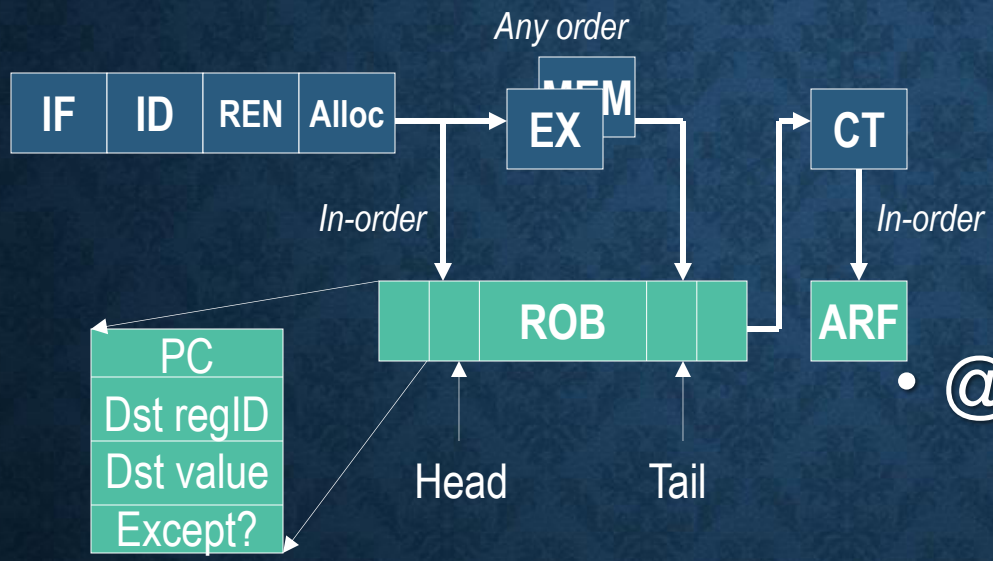
@ **Rename (1 cycle)**

Index table with source operand regID to locate ROB/ARF entry

@ **Alloc**

Allocate ROB entry at Tail

PENTIUM PRO PIPELINE OVERVIEW



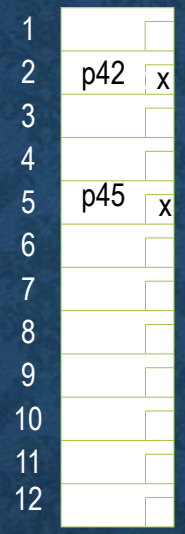
- Reorder Buffer (ROB)
  - Circular queue of spec state
  - May contain multiple definitions of *same* register

- **@ Execute (parallel)**
  - Wait for sources (schedule)
  - Execute instruction (ex)
  - Write back result to ROB
- **@ Commit**
  - Wait until inst @ Head is done
  - If fault, initiate handler
  - Else, write results to ARF
  - Deallocate entry from ROB

REGISTER RENAMING EXAMPLE

Logical Program

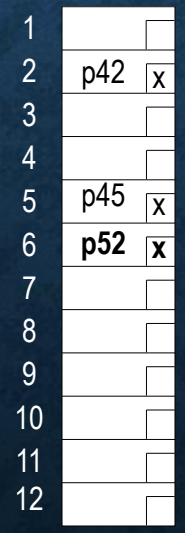
```
r6  = r5 + r2
r8  = r6 + r3
r6  = r9 + r10
r12 = r8 + r6
```



Physical Program

Logical Program

```
r6  = r5 + r2
r8  = r6 + r3
r6  = r9 + r10
r12 = r8 + r6
```



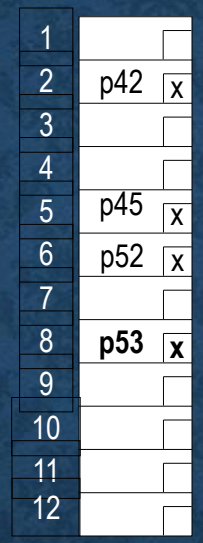
Physical Program

```
p52 = p45 + p42
```

REGISTER RENAMING EXAMPLE

Logical Program

```
r6  = r5 + r2
r8  = r6 + r3
r6  = r9 + r10
r12 = r8 + r6
```

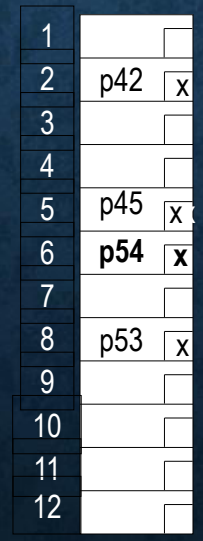


Physical Program

```
p52 = p45 + p42
p53 = p52 + r3
```

Logical Program

```
r6  = r5 + r2
r8  = r6 + r3
r6  = r9 + r10
r12 = r8 + r6
```



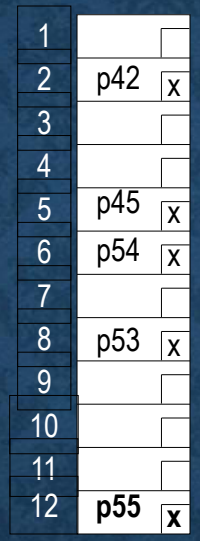
Physical Program

```
p52 = p45 + p42
p53 = p52 + r3
p54 = r9  + r10
```

REGISTER RENAMING EXAMPLE

Logical Program

```
r6  = r5 + r2
r8  = r6 + r3
r6  = r9 + r10
r12 = r8 + r6
```



Physical Program

```
p52 = p45 + p42
p53 = p52 + r3
p54 = r9  + r10
p55 = p53 + p54
```

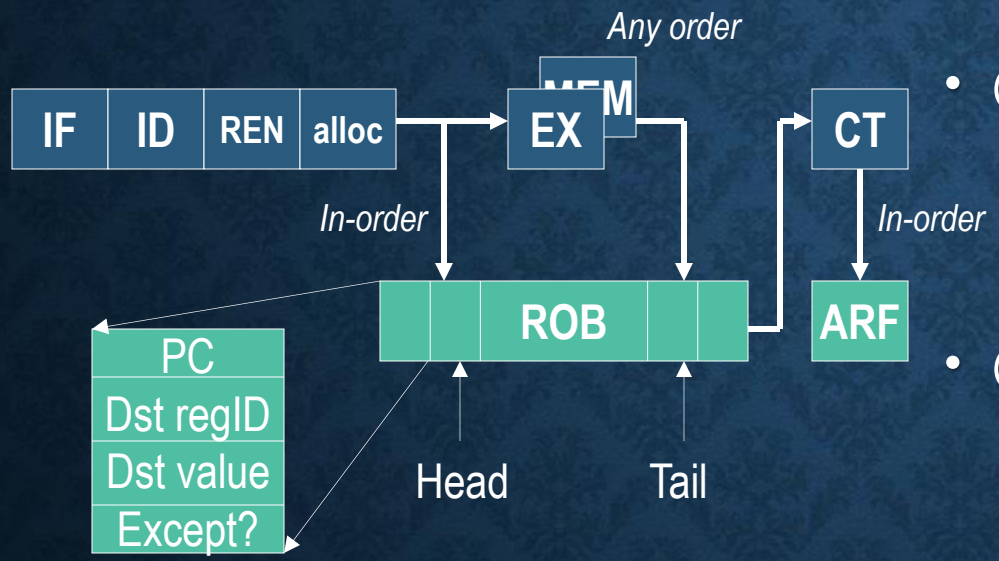
CROSS-CUTTING ISSUE: MISPECULATION

- **What are the impacts of mispeculation or exceptions?**
  - When instructions are flushed from the pipeline, rename mappings must be restored to point-of-restart
  - Otherwise, new instructions will see stale definitions
- Two recovery approaches
  - Simple/slow
    1. Wait until the faulting/mispredicting instruction reaches retirement
    2. Flush *ALL* speculative register definitions by clearing all rename table valid bits
  - Complex/fast
    1. Checkpoint *ENTIRE* rename table anywhere recovery may be needed
    2. At soon as mispeculation detected, recover table associated with PC

DISCUSSION POINTS

- What are the trade-offs between rename table flush recovery and checkpointing?
- What if another instruction (being renamed) needs to access a physical storage entry after it has been overwritten?
- Can I rename memory?

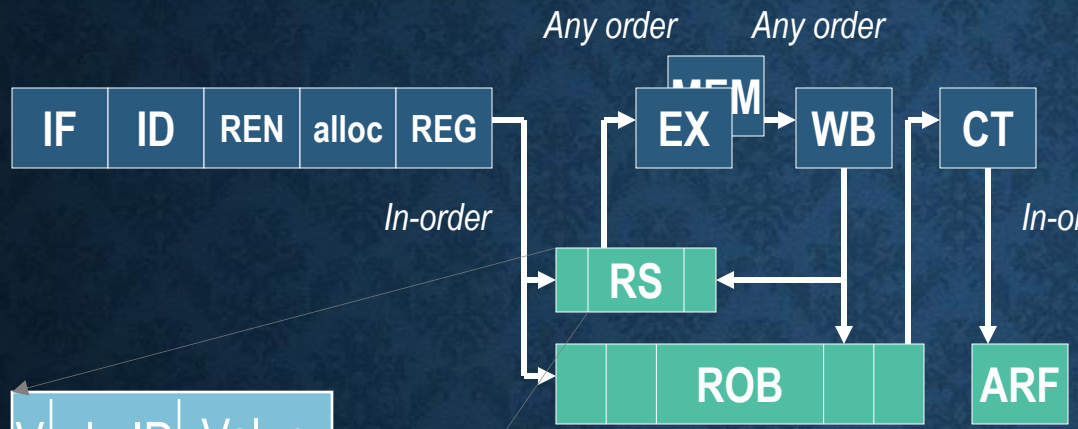
REORDER BUFFER



- Reorder Buffer (ROB)
  - Circular queue of spec state
  - May contain multiple definitions of *same* register

- **@ Alloc**
  - Allocate result storage at Tail
- **@ Execute**
  - Get inputs (ROB T-to-H then ARF)
  - Wait until all inputs ready
  - Execute operation
- **@ WB**
  - Write results/fault to ROB
  - Indicate result is ready
- **@ CT**
  - Wait until inst @ Head is done
  - If fault, initiate handler
  - Else, write results to ARF
  - Deallocate entry from ROB

DYNAMIC INSTRUCTION SCHEDULING



Reservation Stations (RS)

- Associative storage indexed by phyID of dest, returns insts ready to execute
- phyID is ROB index of inst that will compute operand (used to match on broadcast)
- Value contains actual operand
- Valid bits set when operand is available (after broadcast)

@ Alloc

- Allocate ROB storage at Tail
- Allocate RS for instruction

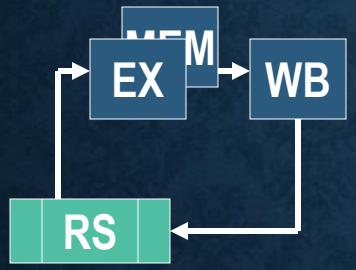
@ REG

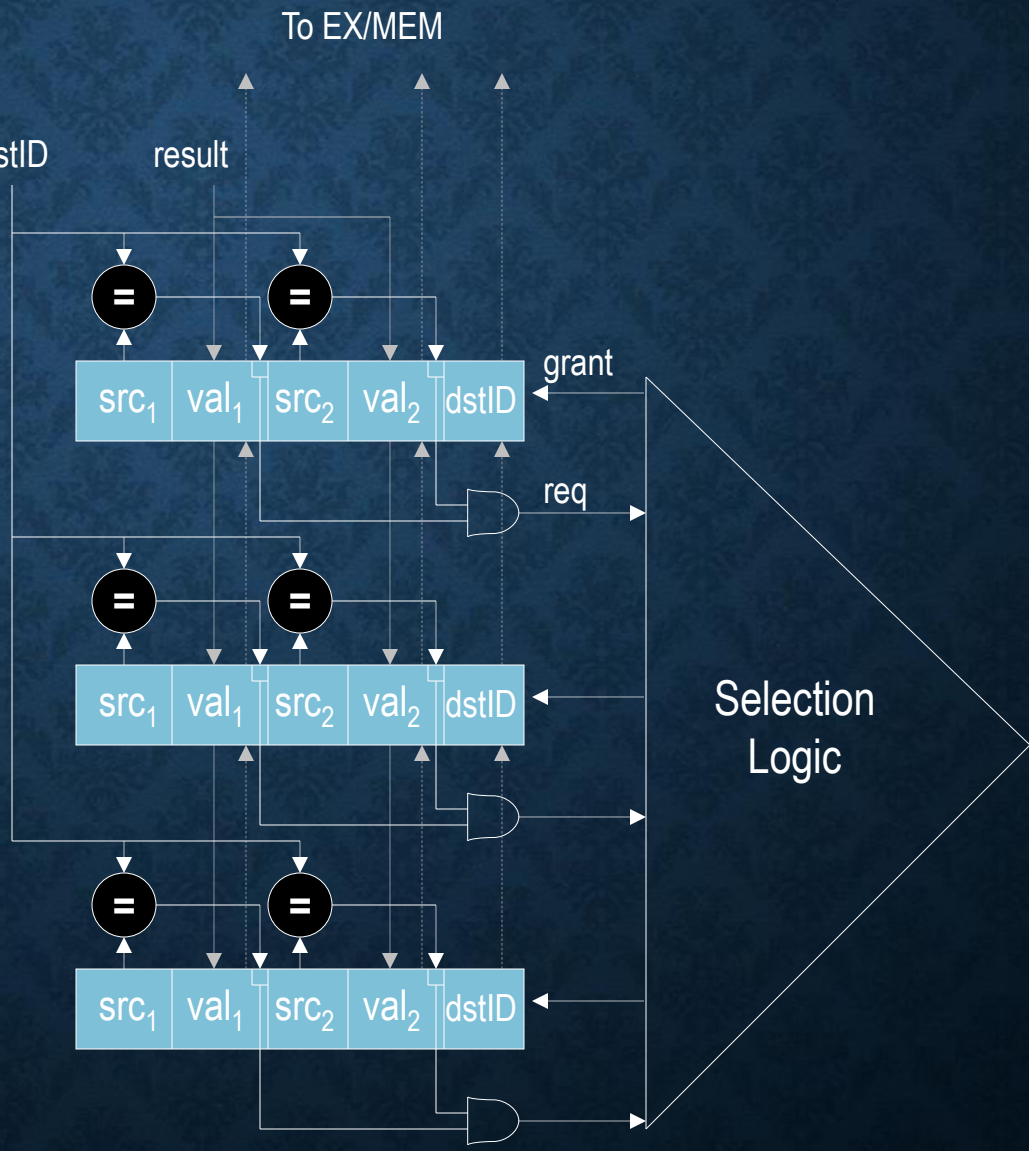
- Get inputs from ROB/ARF entry specified by REN
- Write instruction with available operands into assigned RS

@ WB

- Write result into ROB entry
- Broadcast result into RS with phyID of dest register
- Dellocate RS entry (requires maintenance of an RS free map)

WAKEUP-SELECT-EXECUTE LOOP





WINDOW SIZE VS. CLOCK SPEED

- Increasing the number of RS [Brainiac]
  - Longer broadcast paths
  - Thus more capacitance, and slower signal propagation
  - But, more ILP extracted
- Decreasing the number of RS [Speed Demon]
  - Shorter broadcast paths
  - Thus less capacitance, and faster signal propagation
  - But, less ILP extracted
- Which approach is better and when?

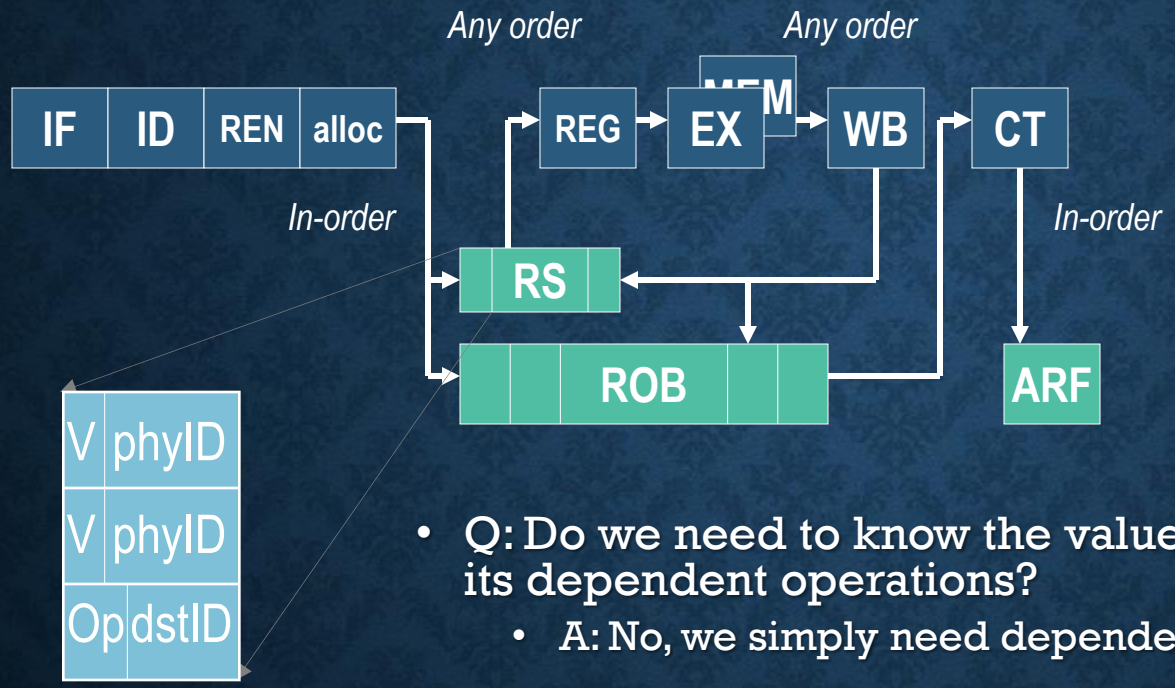
CROSS-CUTTING ISSUE: MISPECULATION

- What are the impacts of mispeculation or exceptions?
  - When instructions are flushed from the pipeline, their RS entries must be reclaimed
  - Otherwise, storage leaks in the microarchitecture
    - This can happen, Alpha 21264 reportedly flushes the instruction window to reclaim all RS resources every million or so cycles
    - The PIII processor reportedly contains a livelock/deadlock detector that would recover this failure scenario
- Typical recovery approach
  - Checkpoint free map at potential fault/mispeculation points
  - Recover the RS free map associated with recovery PC

OPTIMIZING THE SCHEDULER

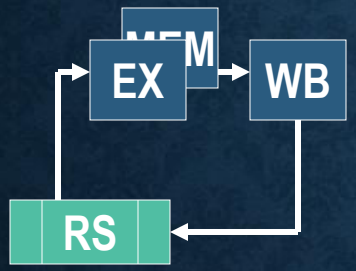
- Optimizing Wakeup
  - Value-less reservation stations
    - Remove register values from latency-critical RS structures
  - Pipelined schedulers
    - Transform wakeup-select-execute loop to wakeup-execute loop
  - Clustered instruction windows
    - Allow some RS to be “close” and other “far away”, for a clock boost
- Optimizing Selection
  - Reservation station banking
    - Associate RS groups with a FU, reduces the complexity of picking

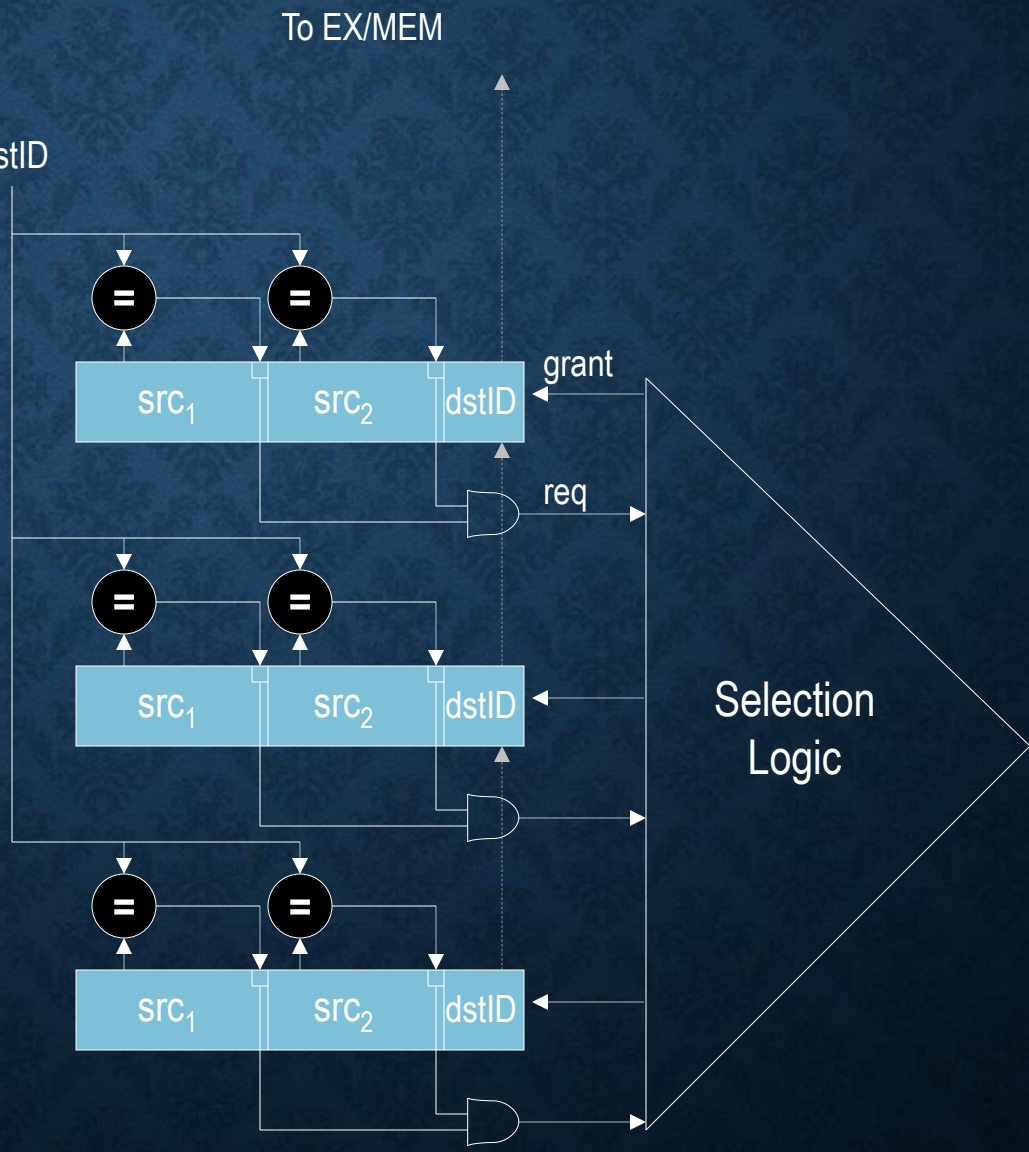
VALUE-LESS RESERVATION STATIONS



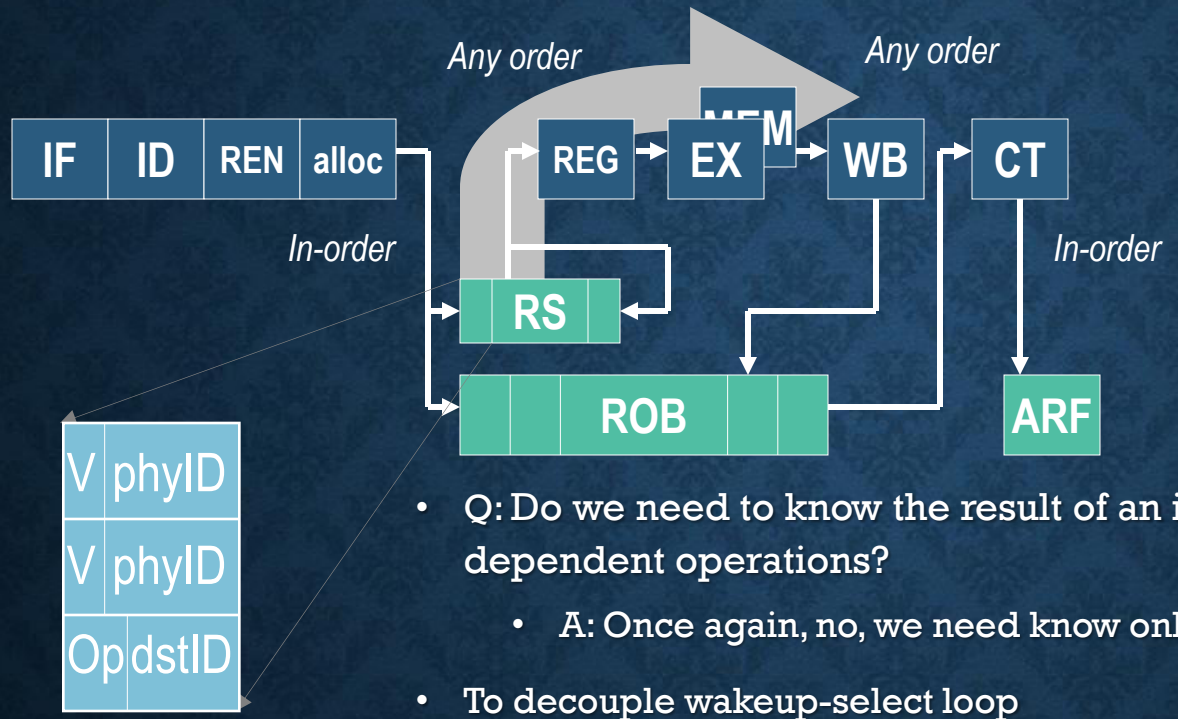
- Q: Do we need to know the value of a register to schedule its dependent operations?
  - A: No, we simply need dependencies and latencies
- Value-less RS only contains required info
  - Dependencies specified by physical register IDs
  - Latency specified by opcode
- Access register file in a later stage, after selection
- Reduces size of RS, which improves broadcast speed

VALUE-LESS RESERVATION STATIONS



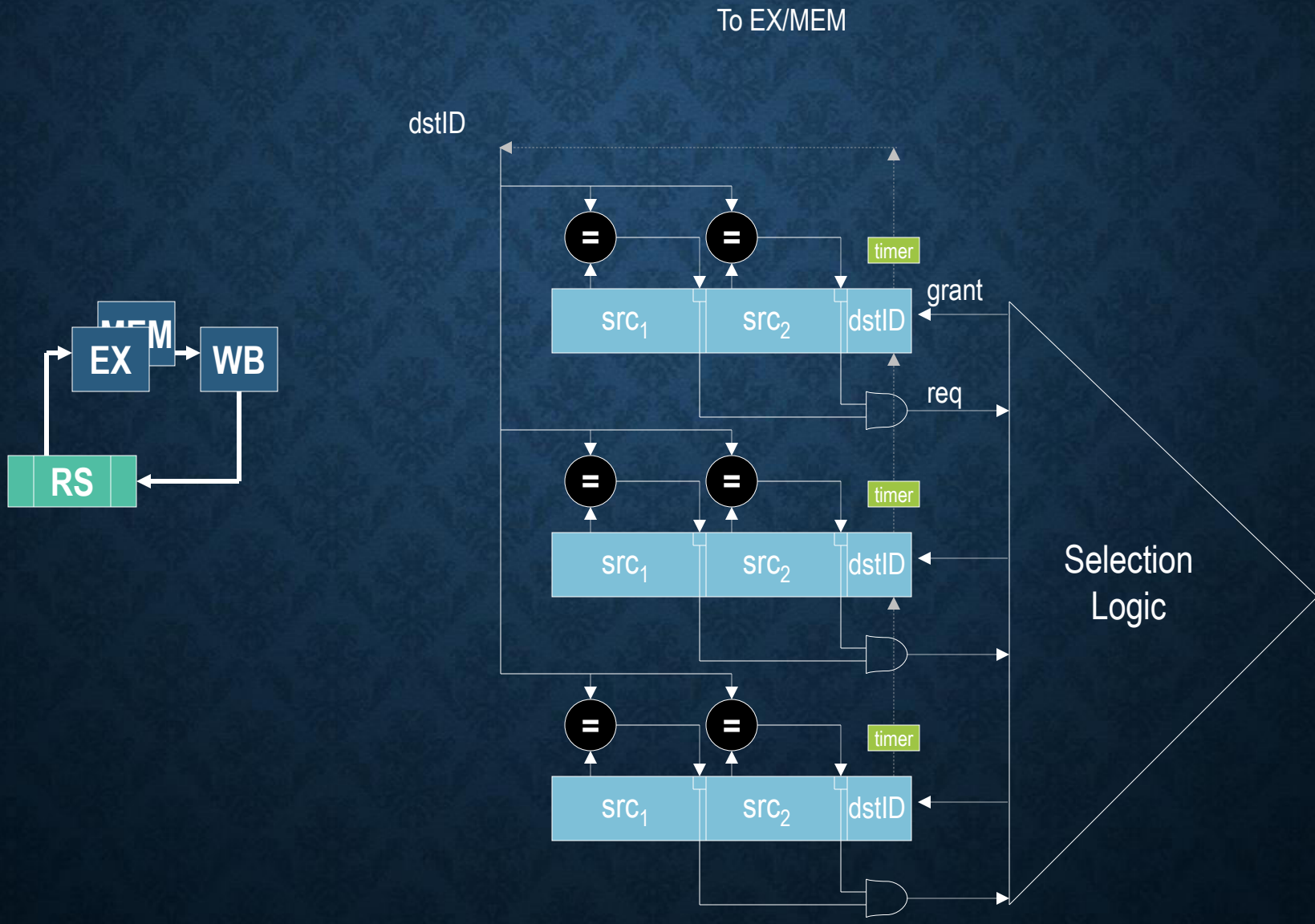


PIPELINED SCHEDULERS

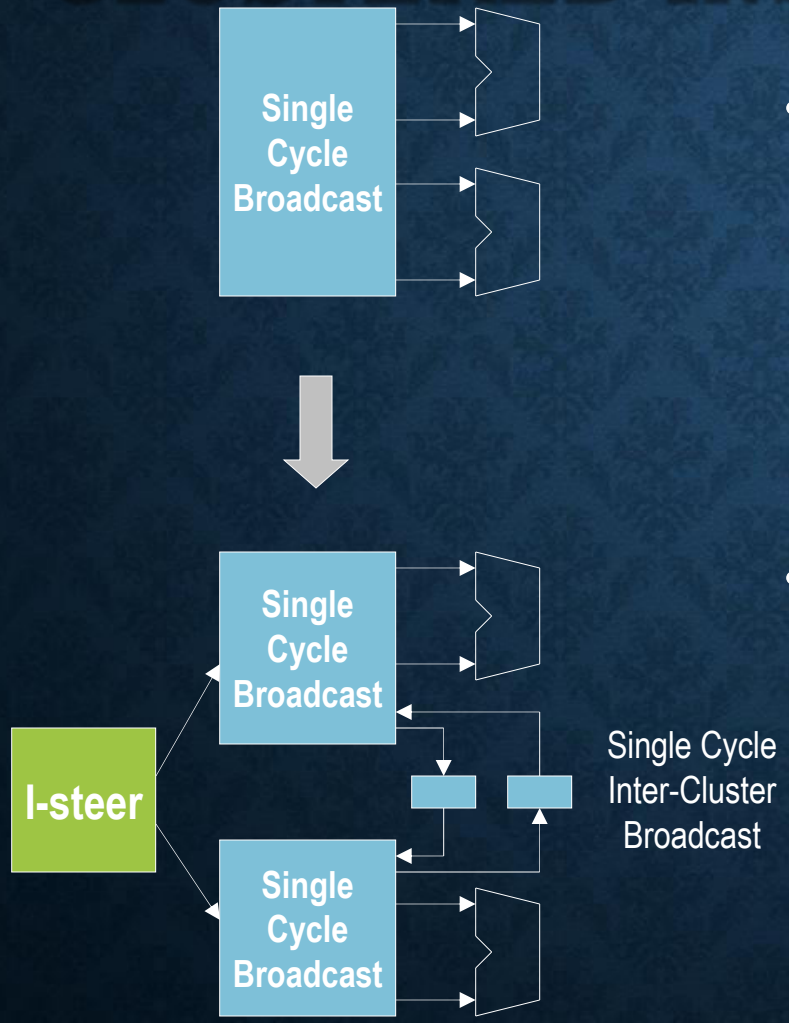


- Q: Do we need to know the result of an instruction to schedule its dependent operations?
  - A: Once again, no, we need know only dependencies and latency
- To decouple wakeup-select loop
  - Broadcast dstID back into scheduler N-cycles after inst enters REG, where N is the latency of the instruction
- What if latency of operation is non-deterministic?
  - E.g., load instructions (2 cycle hit, 8 cycle miss)
  - Wait until latency known before scheduling dependencies (SLOW)
  - Predict latency, reschedule if incorrect
    - Reschedule all vs. selective

PIPELINED SCHEDULERS

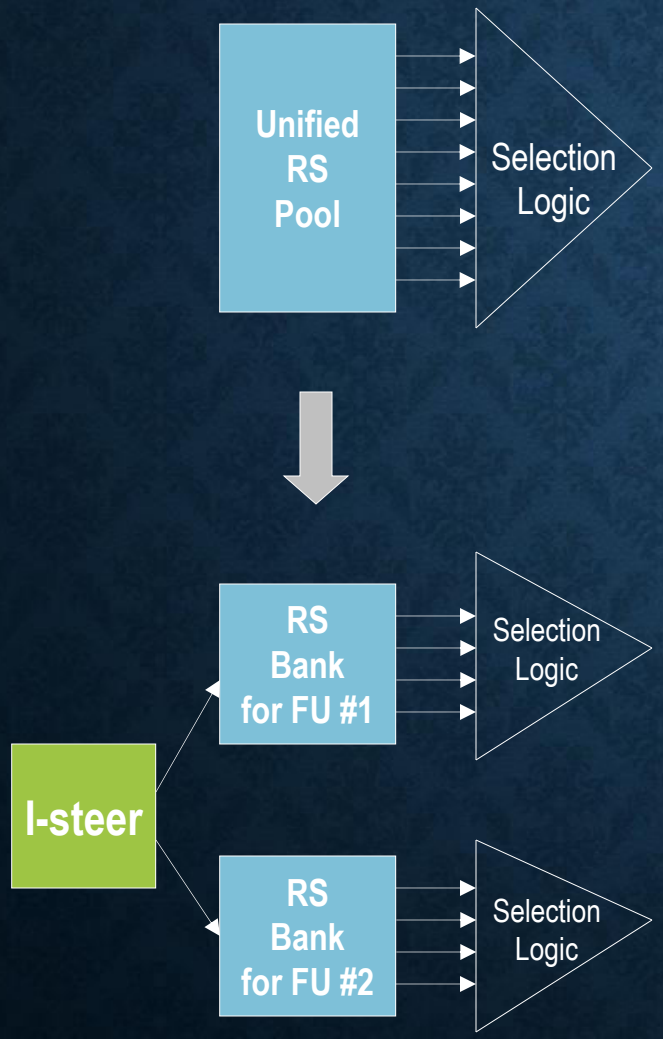


CLUSTERED INSTRUCTION WINDOWS



- Split instruction window into execution clusters
  - W/N RS per cluster, where W is the window size, N is the # of clusters
  - Faster broadcast into split windows
  - Inter-cluster broadcasts take at least an one more cycle
- Instruction steering
  - Minimizes inter-cluster transfers
  - Integer/Floating point split
  - Integer/Address split
  - Dependence-based steering

RESERVATION STATION BANKING



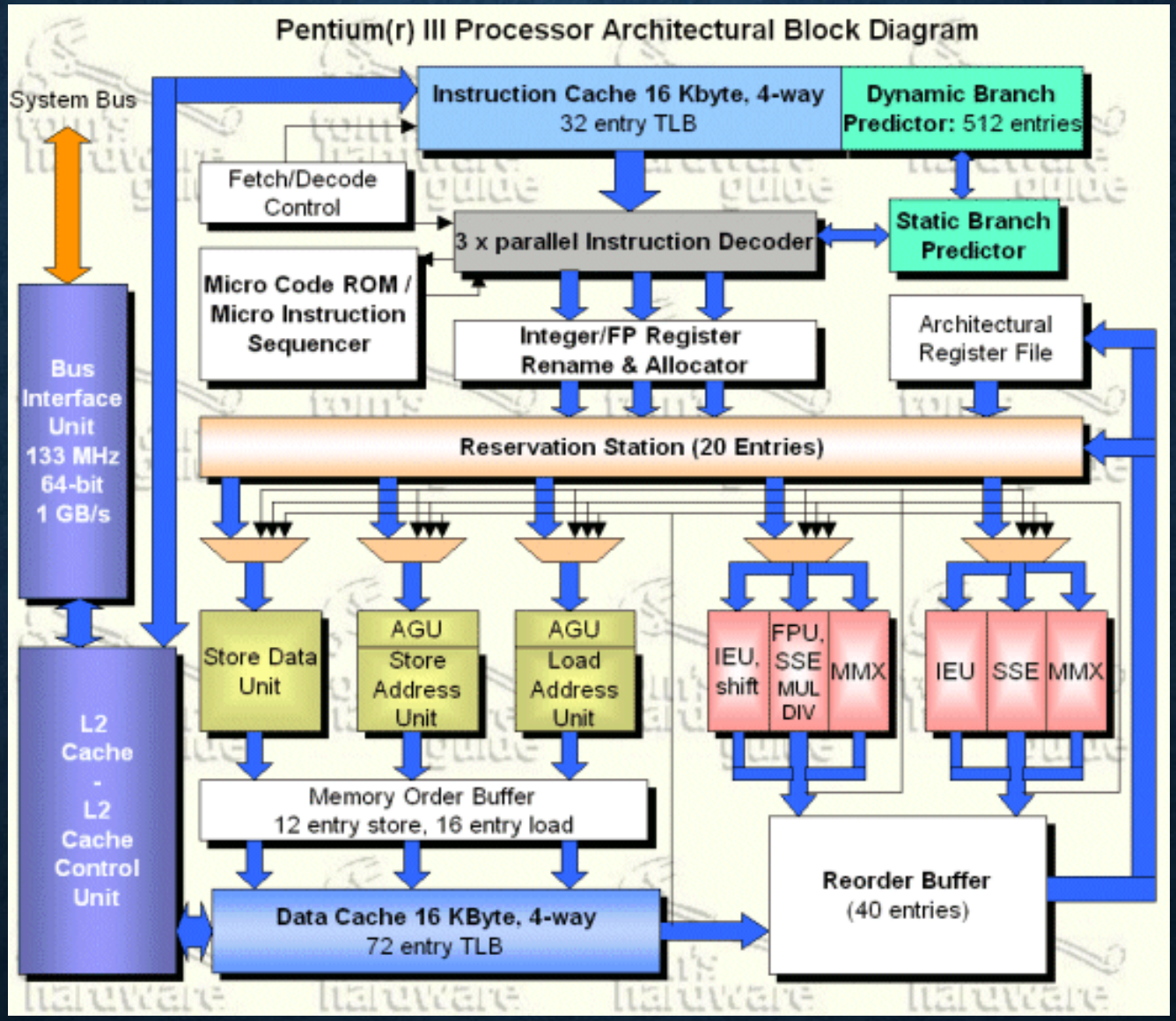
- Split instruction window into banks
  - Group of RS associated with FU
  - Faster selection within bank
- Instruction steering
  - Direct instructions to bank associated with instruction opcode
- Trade-offs with banking
  - Fewer selection candidates speeds selection logic, which is O(log W)
  - But, splits RS resources by FU, increasing the risk of running out of RS resources in ALLOC stage

DISCUSSION POINTS

- If we didn't rename the registers, would the dynamic scheduler still work?
- We can deallocate RS entries out-of-order (which improves RS utilization), why not *allocate* them out-of-order as well?
- What about memory dependencies?

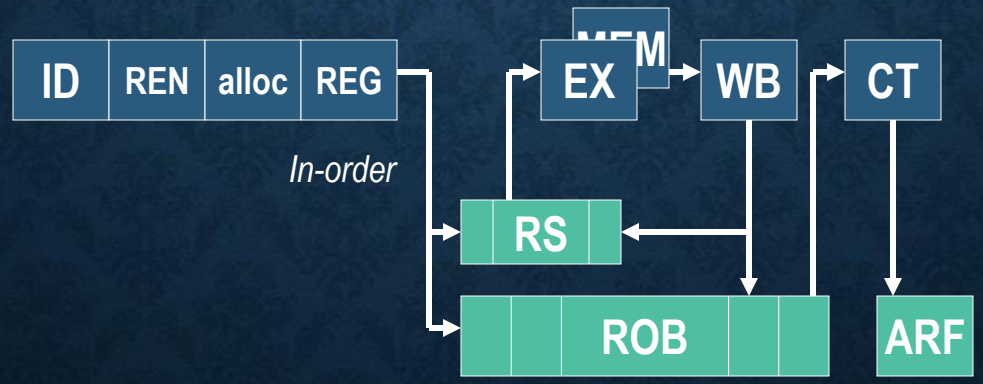
MEMORY DEPENDENCE ISSUES IN AN OUT-OF-ORDER PIPELINE

- Out-of-order memory scheduling
  - Dependencies are known only after address calculation.
  - This is handled in the Memory-order-buffer (MOB)
- When can memory operations be performed out-of-order?
  - What does the MOB have to do to insure that?


Pentium(r) III Processor Architectural Block Diagram
System Bus
Instruction Cache 16 Kbyte, 4-way
32 entry TLB
Dynamic Branch
Predictor: 512 entries
Fetch/Decode
Control
3 x parallel Instruction Decoder
Static Branch
Predictor
Micro Code ROM /
Micro Instruction
Sequencer
Integer/FP Register
Rename & Allocator
Architectural
Register File
Bus
Interface
Unit
133 MHz
64-bit
1 GB/s
Reservation Station (20 Entries)
Store Data
Unit
AGU
Store
Address
Unit
AGU
Load
Address
Unit
IEU,
shift
FPU,
SSE
MUL
DIV
MMX
IEU
SSE
MMX
L2
Cache
-
L2
Cache
Control
Unit
Memory Order Buffer
12 entry store, 16 entry load
Data Cache 16 KByte, 4-way
72 entry TLB
Reorder Buffer
(40 entries)

EFFECTS OF SPECULATION IN AN OUT-OF-ORDER PIPELINE

- What happens when a branch mis-predicts?
  - When should this be recognized?
  - What needs to be cleaned up?



STRUCTURE THAT MUST BE UPDATED AFTER A BRANCH MISPREDICTION.

- ROB
  - Set tail to head to delete everything
- Rename table
  - Mark all entries as invalid (correct values are in the ARF)
- Reservation stations
  - Free all reservation station entries
- MOB
  - Free all MOB entries
  - Correctly handle any outstanding memory operations.