CASPER

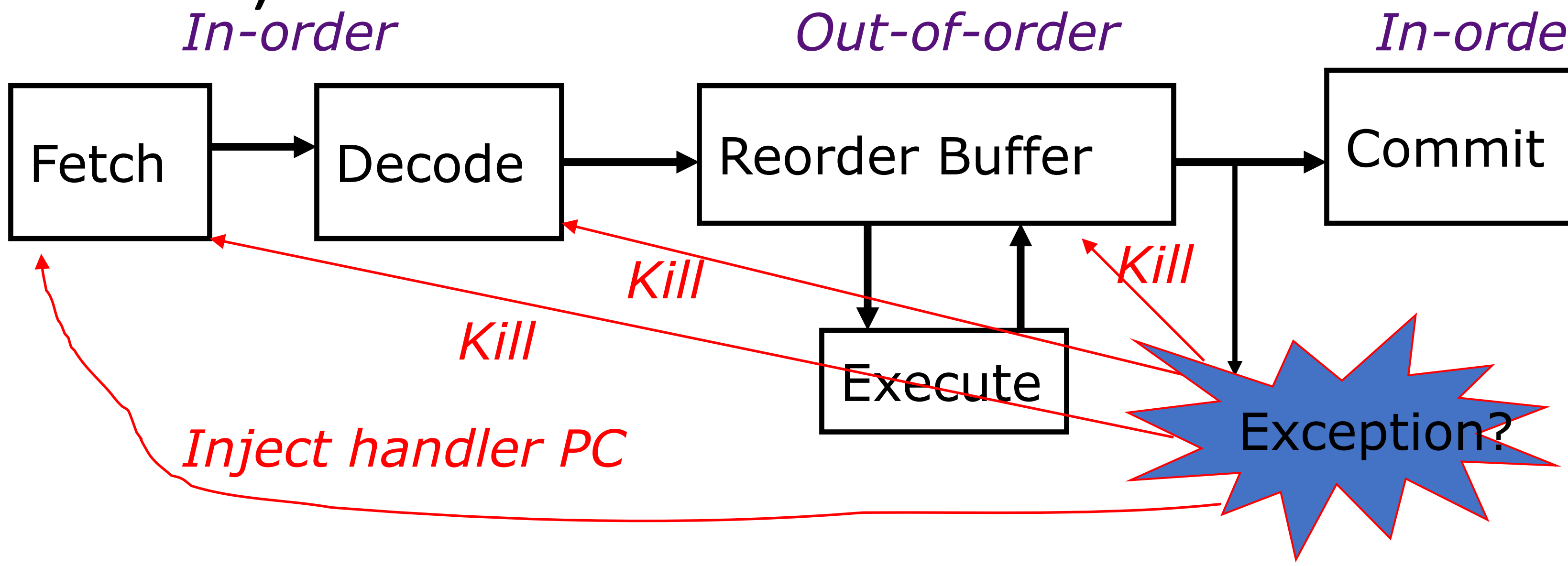
CS305: Computer Architecture

Speculative Execution

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

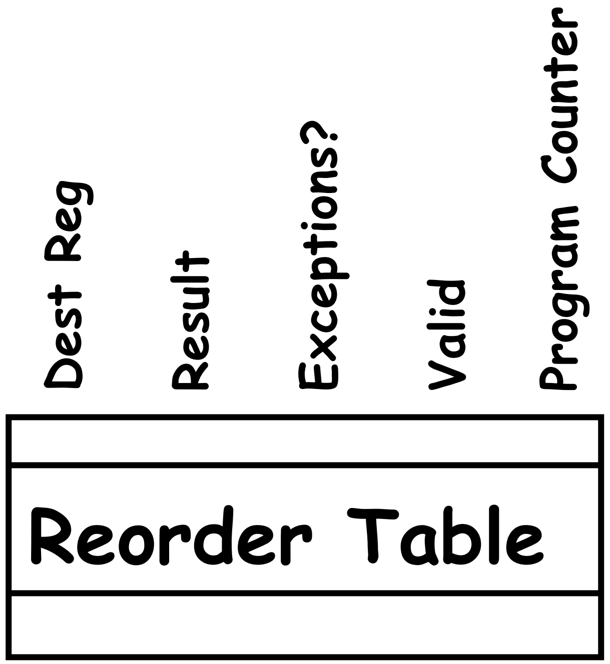
Tomasulo, O3 completion, we need inorder complete (commit)☹

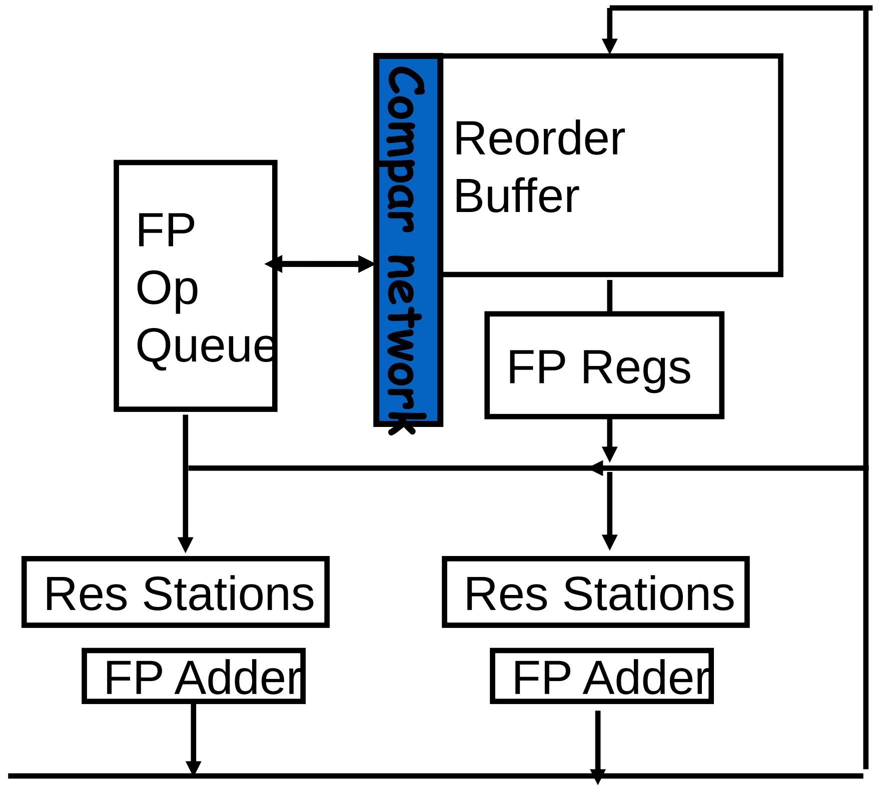


- Instructions fetched and decoded into instruction reorder buffer in-order
- Execution is out-of-order ($\Rightarrow$ out-of-order completion)
- *Commit* (write-back to architectural state, i.e., regfile & memory) is in-order

Temporary storage needed to hold results before commit (shadow registers and store buffers)

Dynamic scheduling with speculative execution





Need as many ports on ROB as register file

Speculative O3 with Tomasulo and ROB

1. Issue—get instruction from FP Op Queue

   If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination (this stage sometimes called "dispatch")

2. Execution—operate on operands (EX)

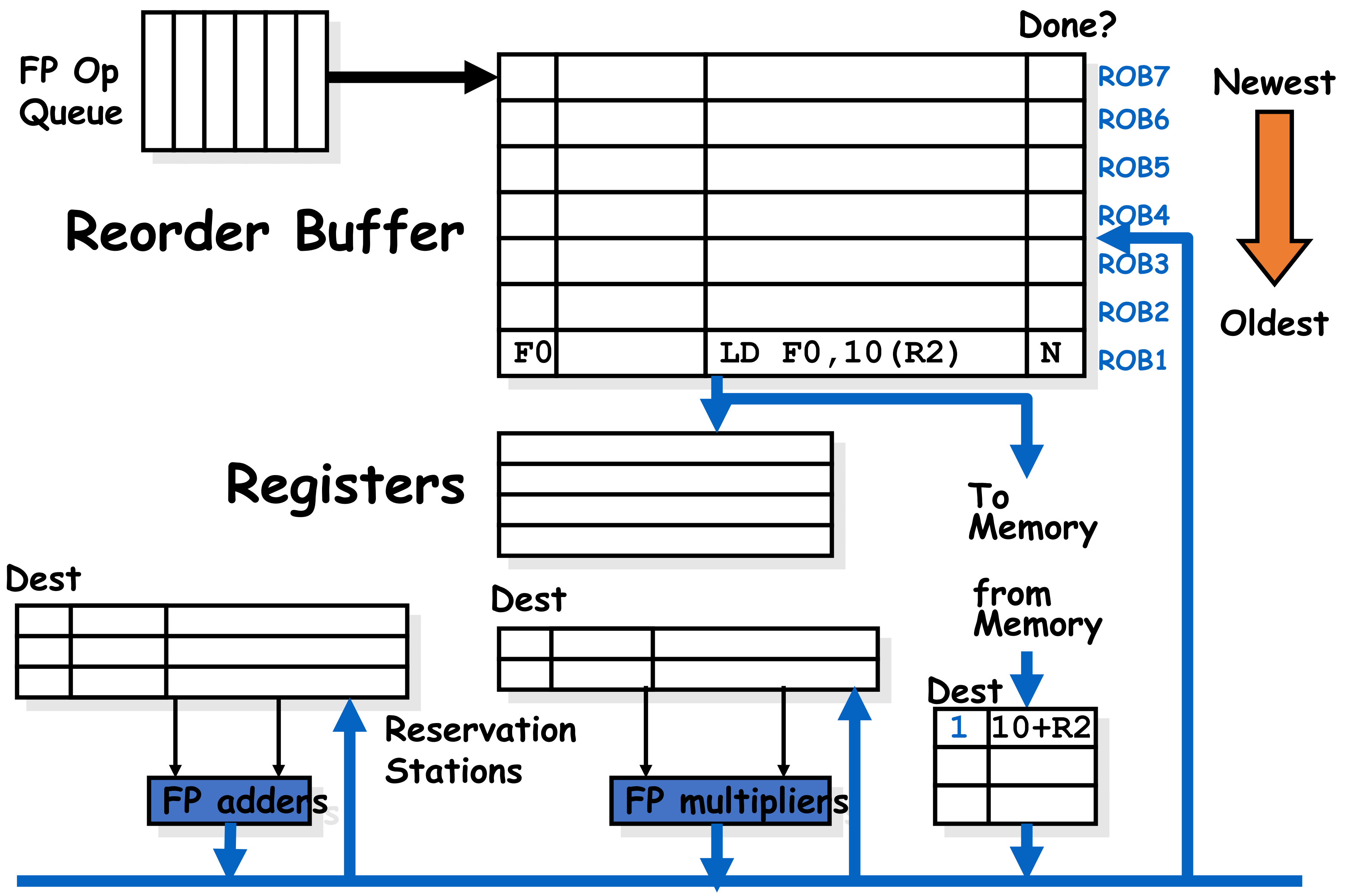
   When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute; checks RAW (sometimes called "issue")

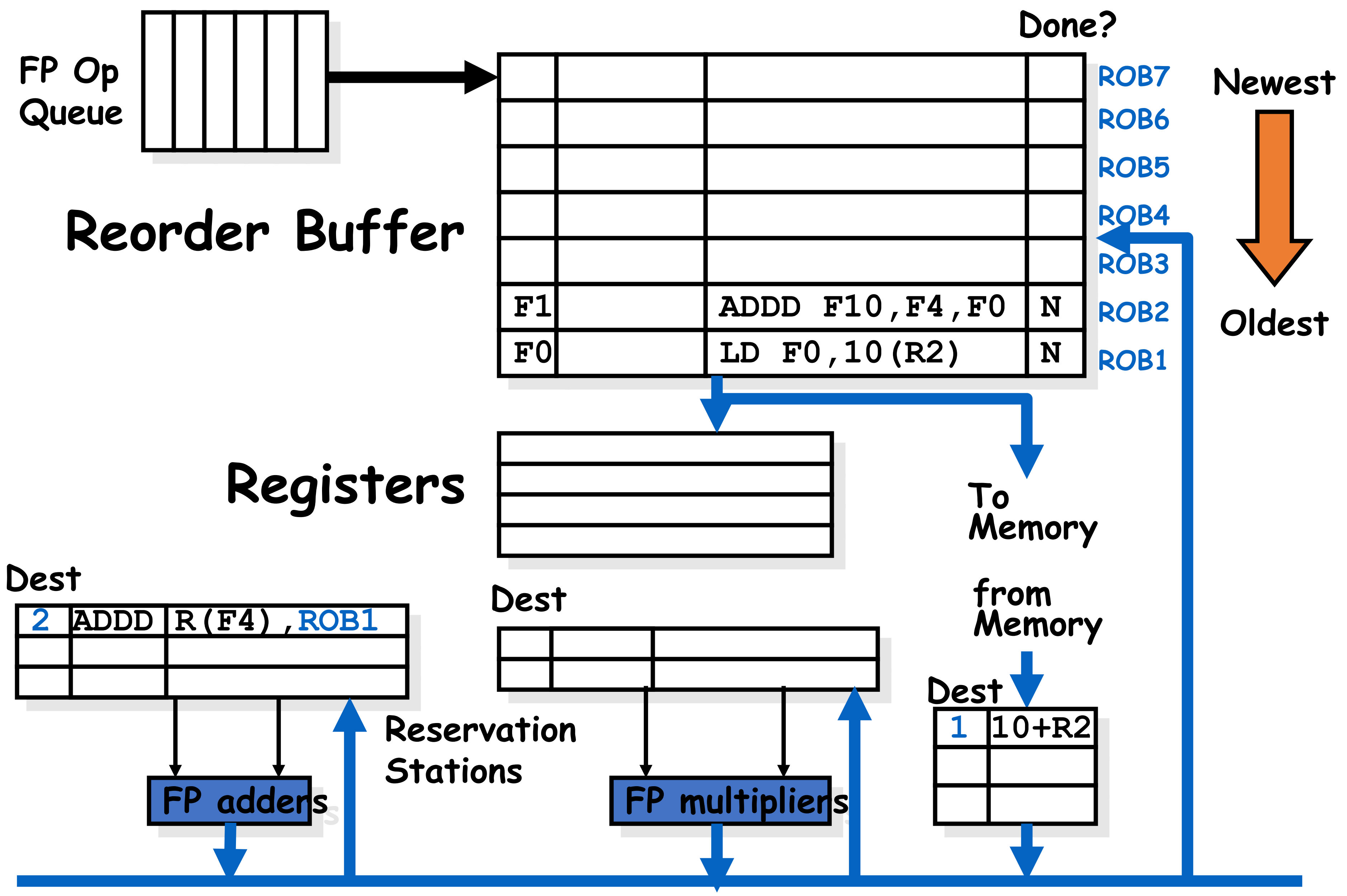
3. Write result—finish execution (WB)

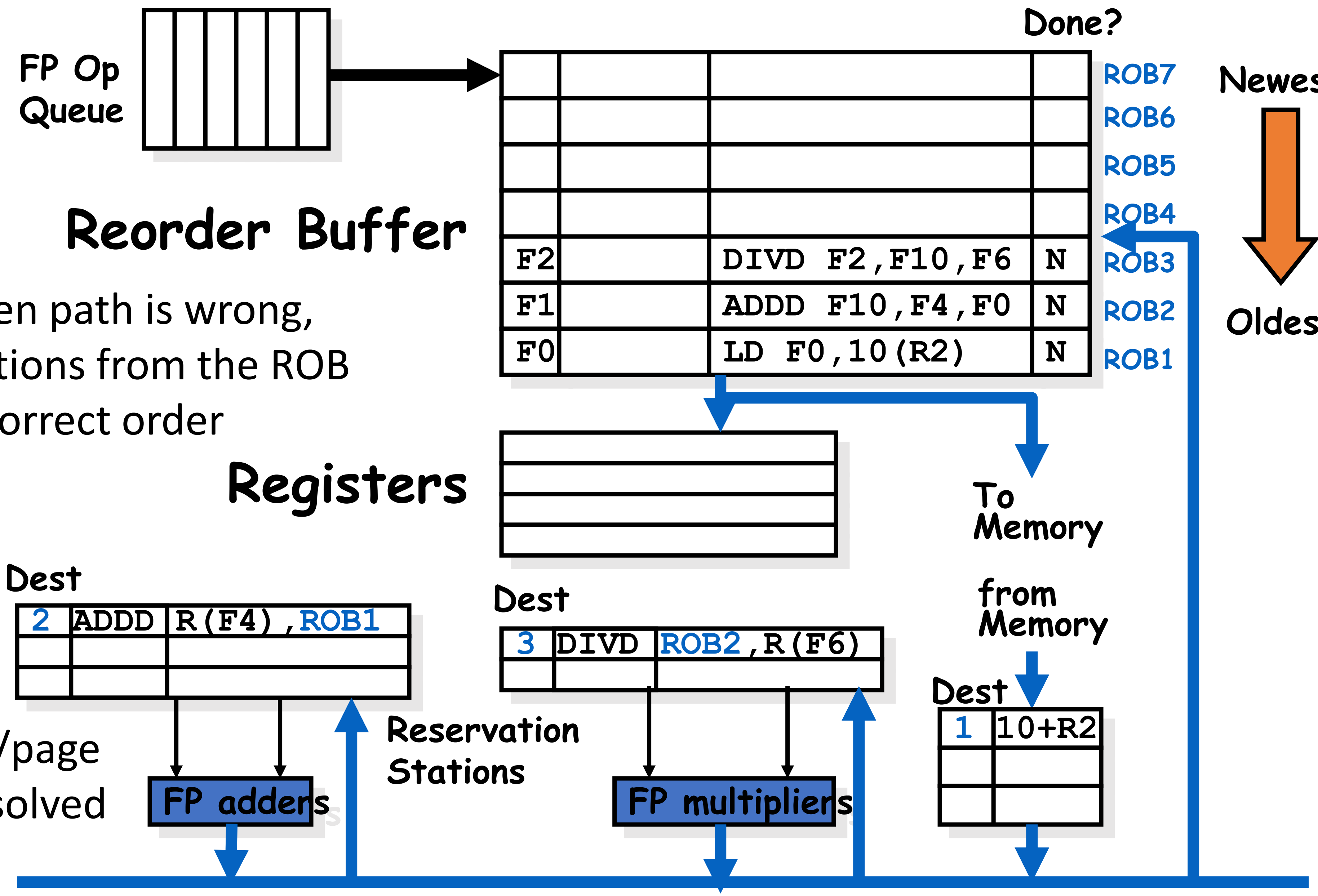
   Write on Common Data Bus to all awaiting FUs
   & reorder buffer; mark reservation station available.

4. Commit—*When instruction reaches head of the ROB,* update register with reorder result

   When instr. at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer. Mispredicted branch flushes reorder buffer (sometimes called "graduation")


FP Op Queue
Done?
Newest
Oldest
Reorder Buffer
ROB7
ROB6
ROB5
ROB4
ROB3
ROB2
ROB1
F0
LD F0,10(R2)
N
Registers
To Memory
from Memory
Dest
Dest
Dest
1
10+R2
Reservation Stations
FP adders
FP multipliers


FP Op Queue
Done?
Newest
Oldest
Reorder Buffer
ROB7
ROB6
ROB5
ROB4
ROB3
F1 | ADDD F10,F4,F0 | N | ROB2
F0 | LD F0,10(R2) | N | ROB1
Registers
To Memory
from Memory
Dest
2 | ADDD | R(F4),ROB1
Dest
Dest
1 | 10+R2
Reservation Stations
FP adders
FP multipliers



If the predicted taken path is wrong, flush out all instructions from the ROB and reissue in the correct order

Remember exception/page fault handling gets resolved when the instruction reaches the head of the ROB

Memory Disambiguation

- Question: Given a load that follows a store in program order, are the two related?
  - (Alternatively: is there a RAW hazard between the store and the load)?

Eg:

```
st   0(R2),R5
ld   R6,0(R3)
```

- Can we go ahead and start the load early?
  - Answer is that we are not allowed to start load until we know that address 0(R2) $\neq$ 0(R3)

A Complex world

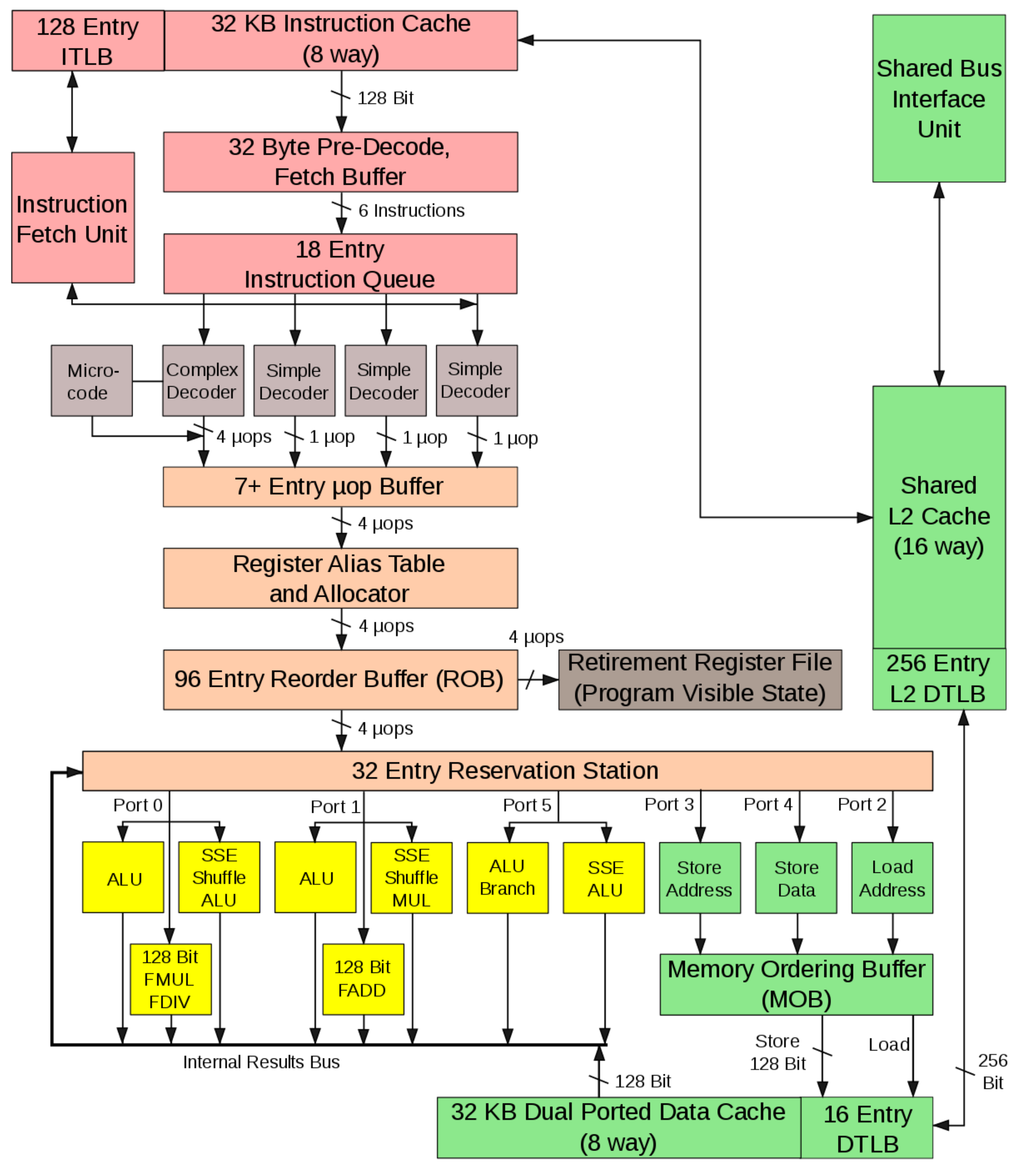
If we start fetching multiple instructions in one go; multiple issue;

All the structures should be updated concurrently for multiple instructions ☹

Modern processors use pipelined structures. Updates at the rising/falling edges.

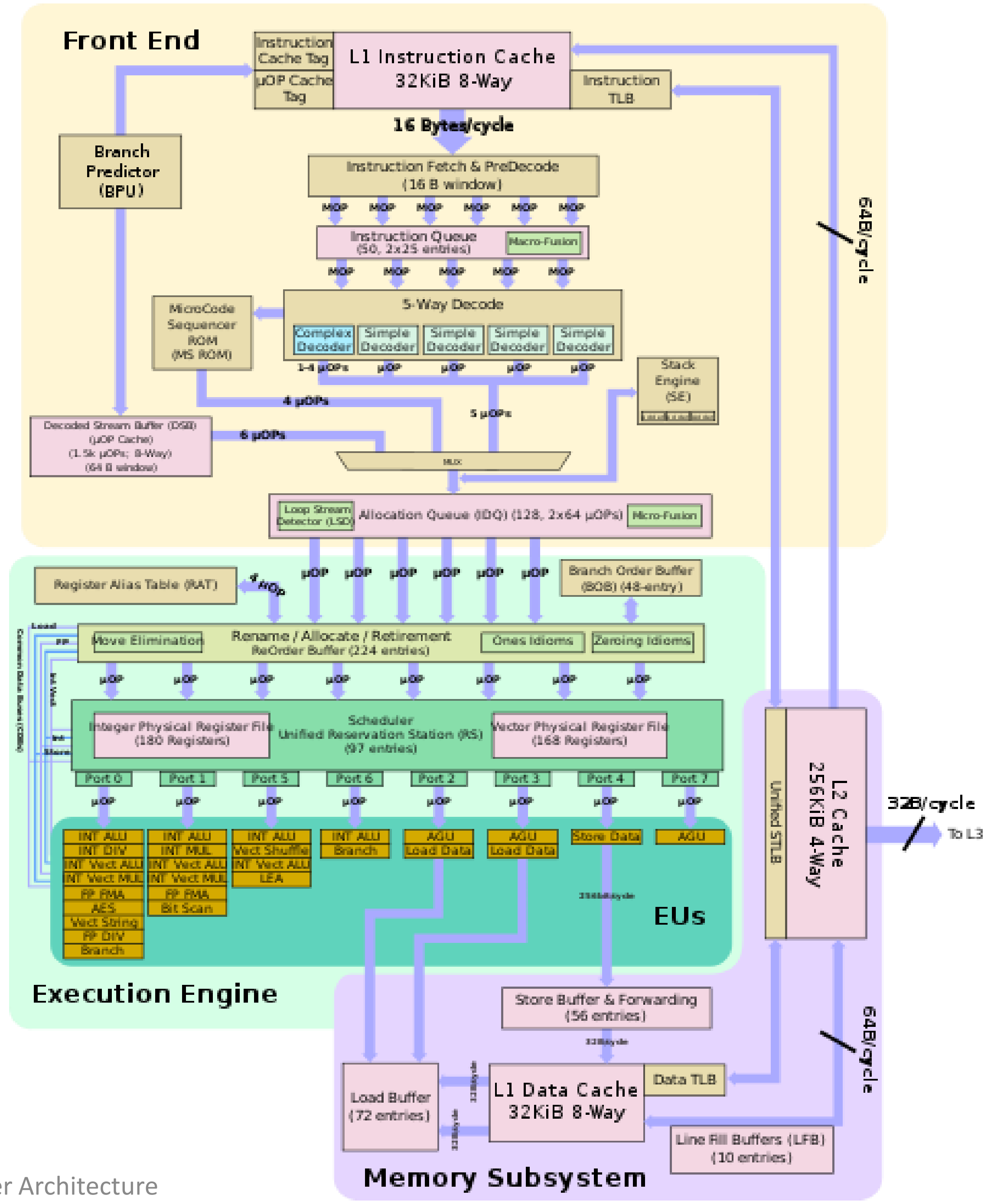
Is this out-dated stuff

Intel Core Microarchitecture



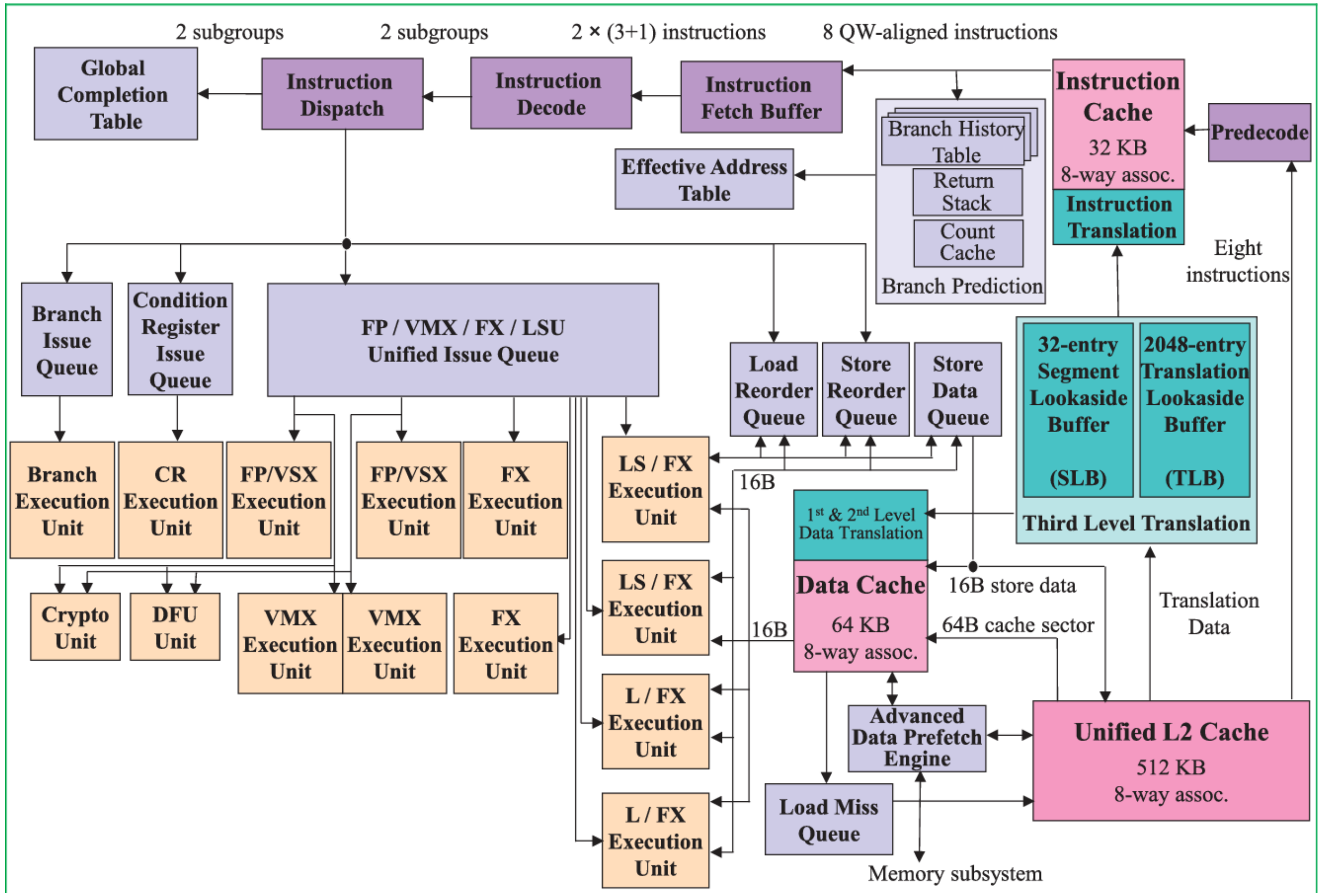
Intel Core 2 Architecture

Some more

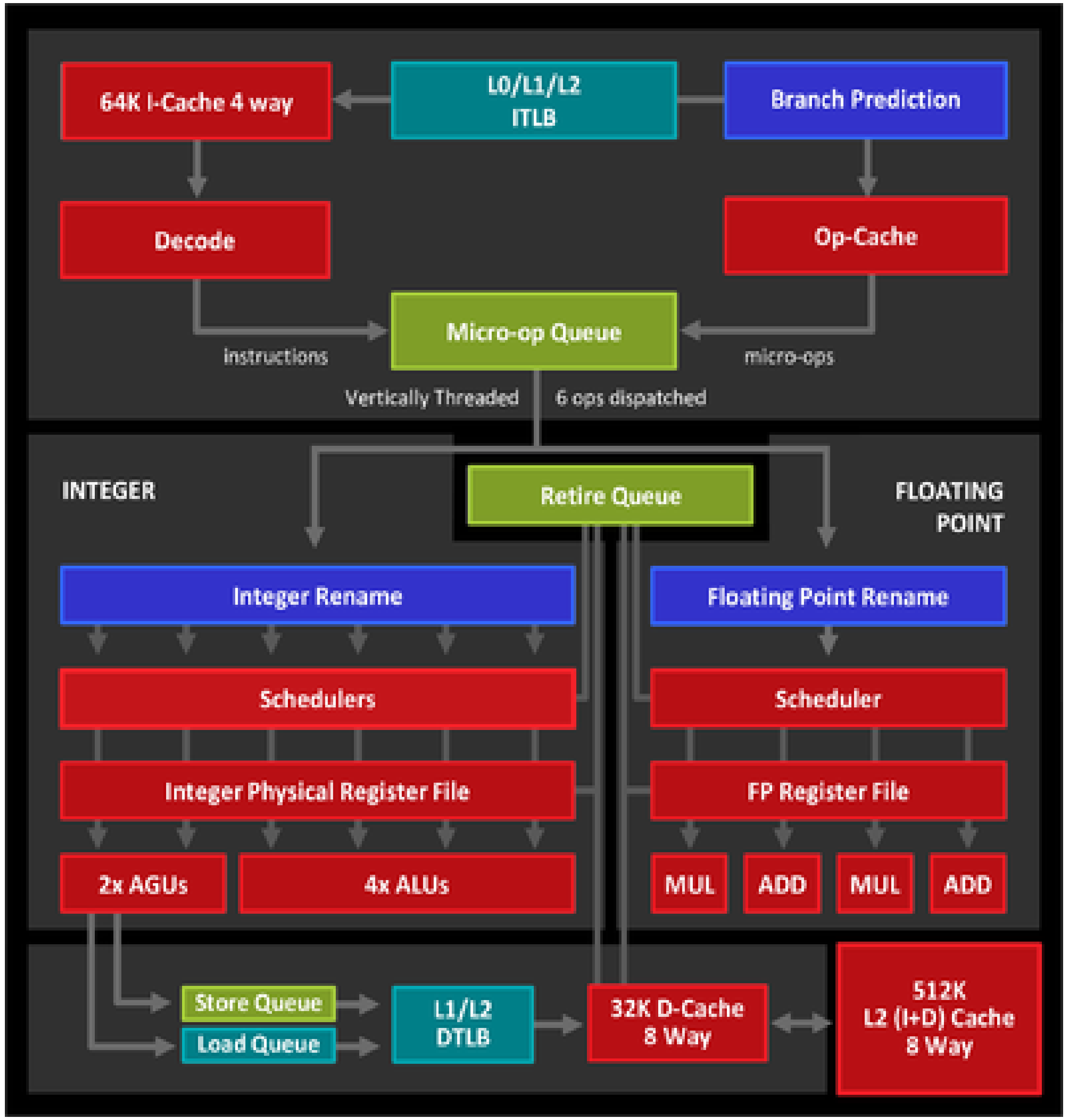


Front End
Instruction Cache Tag
µOP Cache Tag
L1 Instruction Cache 32KiB 8-Way
Instruction TLB
16 Bytes/cycle
Branch Predictor (BPU)
Instruction Fetch & PreDecode (16 B window)
MOP
Instruction Queue (50, 2x25 entries)
Macro-Fusion
5-Way Decode
MicroCode Sequencer ROM (MS ROM)
Complex Decoder
Simple Decoder
1-4 µOPs
µOP
4 µOPs
5 µOPs
6 µOPs
Stack Engine (SE)
Decoded Stream Buffer (DSB) (µOP Cache) (1.5k µOPs; 8-Way) (64 B window)
MUX
Loop Stream Detector (LSD)
Allocation Queue (IDQ) (128, 2x64 µOPs)
Micro-Fusion
64B/cycle
Register Alias Table (RAT)
4 µOPs
Branch Order Buffer (BOB) (48-entry)
Move Elimination
Rename / Allocate / Retirement ReOrder Buffer (224 entries)
Ones Idioms
Zeroing Idioms
Common Data Buses (CDBs)
Integer Physical Register File (180 Registers)
Scheduler Unified Reservation Station (RS) (97 entries)
Vector Physical Register File (168 Registers)
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
INT ALU
INT DIV
INT Vect ALU
INT Vect MUL
FP FMA
AES
Vect String
FP DIV
Branch
INT MUL
Bit Scan
Vect Shuffle
LEA
AGU
Load Data
Store Data
EUs
Execution Engine
Unified STLB
L2 Cache 256KiB 4-Way
32B/cycle
To L3
Store Buffer & Forwarding (56 entries)
Load Buffer (72 entries)
L1 Data Cache 32KiB 8-Way
Data TLB
Line Fill Buffers (LFB) (10 entries)
Memory Subsystem

Some from IBM



AMD


64K I-Cache 4 way
L0/L1/L2 ITLB
Branch Prediction
Decode
Op-Cache
Micro-op Queue
instructions
micro-ops
Vertically Threaded
6 ops dispatched
INTEGER
Retire Queue
FLOATING POINT
Integer Rename
Floating Point Rename
Schedulers
Scheduler
Integer Physical Register File
FP Register File
2x AGUs
4x ALUs
MUL
ADD
MUL
ADD
Store Queue
Load Queue
L1/L2 DTLB
32K D-Cache 8 Way
512K L2 (I+D) Cache 8 Way

Look for

For Example,

https://en.wikichip.org/wiki/intel

https://ark.intel.com/content/www/us/en/ark.html#@Processors

Doh Jeh