Spring 2025 Final Exam Review

When / Where

Thursday, 8 May 2025 7:30–9:30 (7:30 **AM** – 9:30 **AM**) CDT.

Room 1212 Patrick F. Taylor Hall.

Conditions

Closed Book, Closed Notes

Can bring one $215 \times 280\,\mathrm{mm}$ note sheet.

Cannot use communication devices.

Format

Two or three or maybe four medium to long problems.

Short-answer questions.

Resources

Check course Web page daily for hints, new resources.

Web page: `https://www.ece.lsu.edu/ee4720/index.html`

RSS feed: `https://www.ece.lsu.edu/ee4720/rss_home.xml`

Solved tests and homework:. `https://www.ece.lsu.edu/ee4720/prev.html`

Study Recommendations

Study homework assigned this semester—**test questions are often based on homework questions.**

$\boxed{\textit{\underline{Solve}}\ \textbf{previous test problems}}$, start with more recent problems.

Memorizing solutions is not the same as solving problems.

Following and understanding solutions is not the same as solving problems.

Use the solutions for brief hints and to check your own solutions.

Emphases

Implementation Diagrams and Pipeline Execution Diagrams

They are a *team*, so study them together.

Designing and analyzing control logic and datapath.

Hardware, Including Control Logic

We can't rely on magic performed by others.

Instruction Use

Should be able to easily write MIPS programs.

Should be able to use MIPS instructions in examples.

Not required to memorize instruction names, except for common MIPS instructions.

Topics

Introductory Material

ISA v. Implementation.

Benchmark types.

Compiling and Optimization

SPECcpu Benchmark Suite

See Lecture Slides 2—`https://www.ece.lsu.edu/ee4720/2025/lsli02.pdf`,
SPECcpu description—`http://www.spec.org/benchmarks.html`,
and the SPECcpu run and reporting rules—`http://www.spec.org/cpu2006/Docs/runrules.html`

What SPECcpu is supposed to measure. (Potential of new CPU implementations.)

Importance of having tester compile code.

Peak v. Base tuning: code preparation, use of results.

Why should we trust the SPEC rules?

Why should we trust the results of a test?

Benchmark programs (types, how they were selected).

Compilers and Optimization

Steps in building and compiling.

Basic optimization techniques, compiler optimization switches.

Profiling.

Instruction Coding.

Fixed-length, variable-length, and bundled instructions.

Splitting of opcode field (as in MIPS type-R instructions).

ISA Classifications: RISC, CISC, VLIW

Dependency Definitions

Hazard Definitions

ISA Familiarity

MIPS

Read programs, write programs, implement (design hardware)

SPARC

Read common instructions.

Use of condition codes.

Instruction coding differences.

RISC-V

Instruction coding differences.

MIPS

Classification: RISC

Goals: ISA should allow simple, high-speed implementation.

Instruction types.

Know how to read and write MIPS programs.

Statically Scheduled MIPS Implementations

See Lecture Slides 6—`https://www.ece.lsu.edu/ee4720/2025/lsli06.pdf`
and Statically Scheduled Study Guide—`https://www.ece.lsu.edu/ee4720/guides/ssched.pdf`

Pipelined Implementations

Basic (no bypassing, 2-cycle branch penalty), bypassed, branch in ID.

For a Given Pipelined Implementation

Show pipeline execution diagrams.

Show register contents at any cycle.

Design control logic.

Add logic and datapath to provide new bypass, insn, etc.

Determine instruction throughput (IPC).

Long Latency (FP) Operations

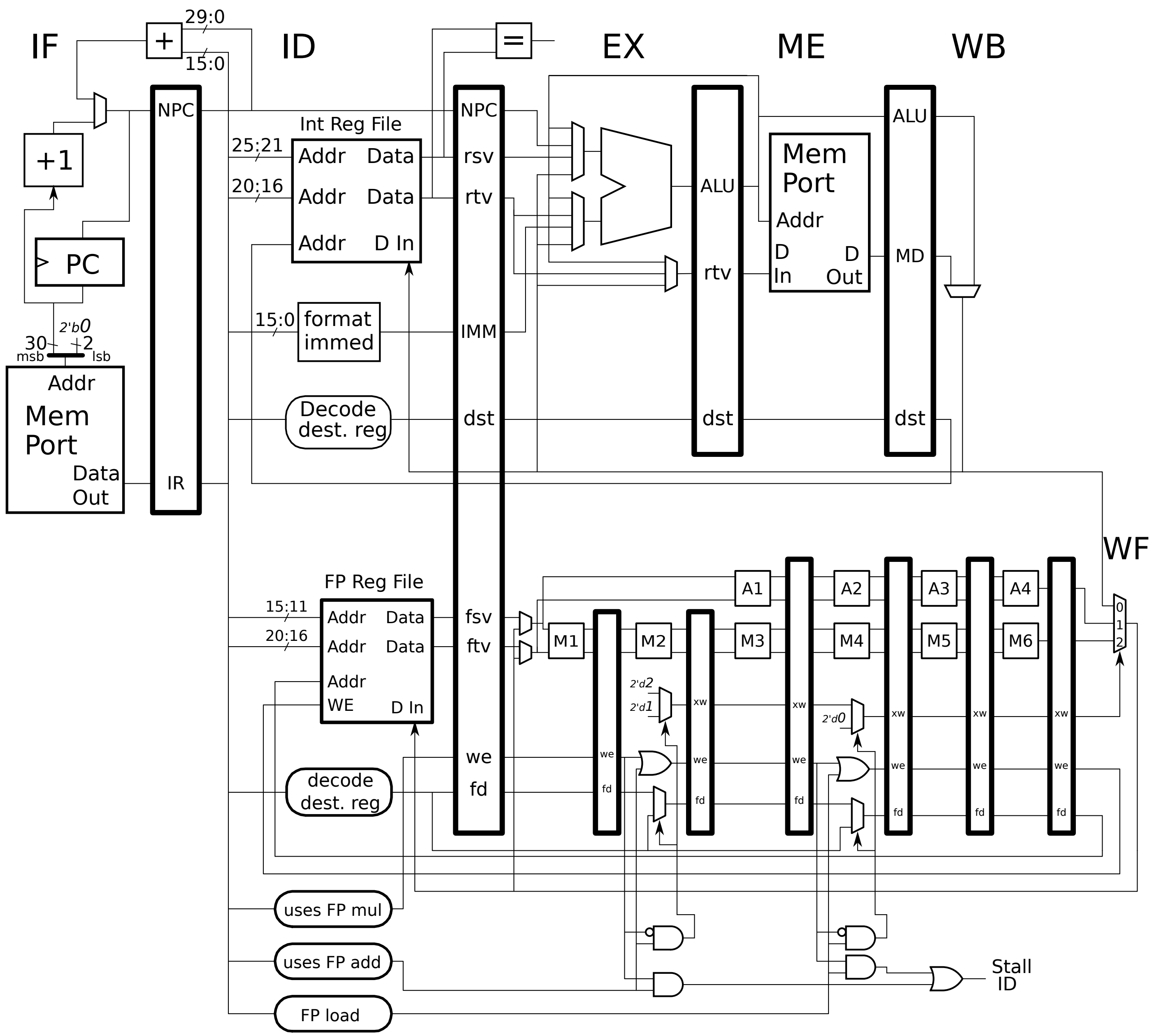
Types of operations. (Floating point and maybe load.)

Degree of pipelining: Initiation interval.

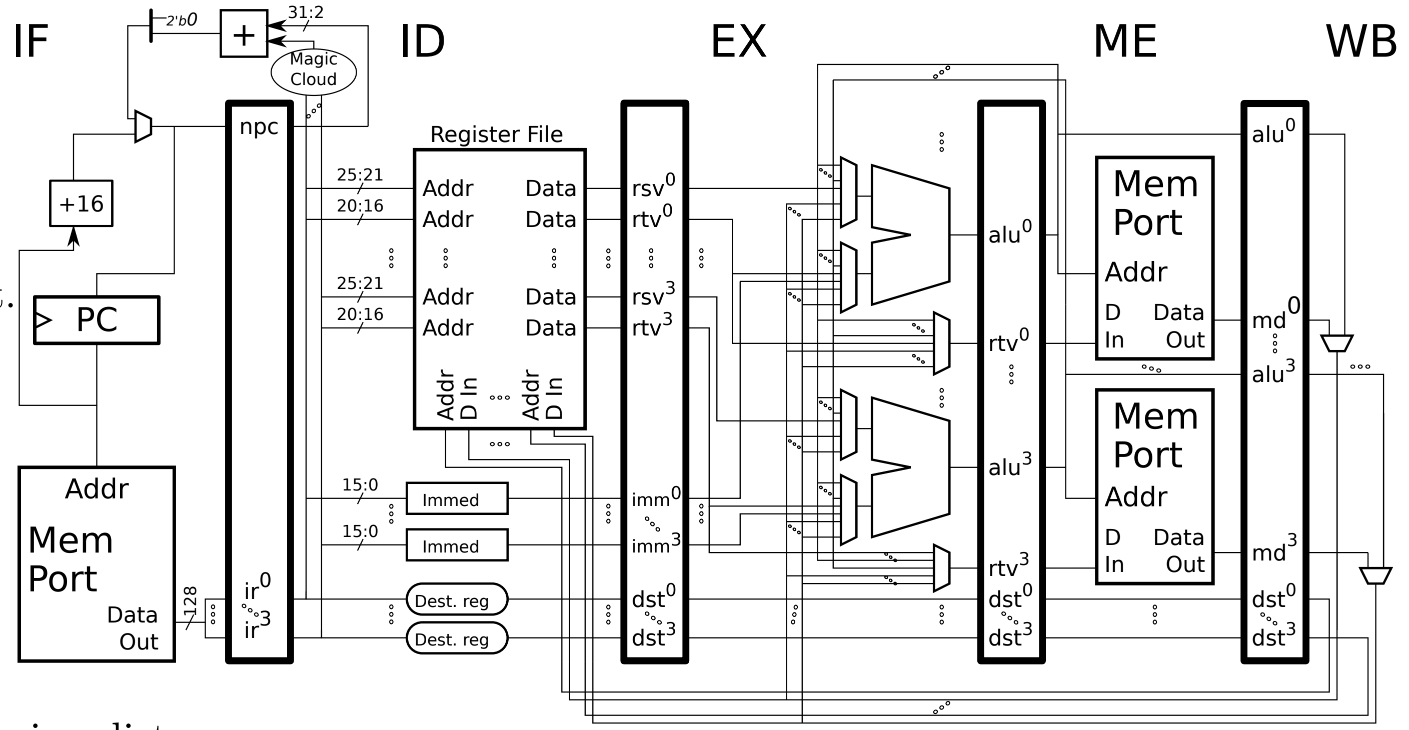
Detecting functional unit structural hazards.

Detecting WB structural hazards: pipeline control logic.

Handling WAW hazards.



Superscalar and VLIW



n-Way Superscalar

Duplication of Resources.

Duplicated $n\times$: Fetch, decode, rename, writeback, commit.

Duplicated $< n\times$: load/store, floating-point units.

Costs: $\propto n$ functional units; $\propto n^2$ bypass, control.

Performance Limiters

Limiters due to device technology: Lower clock with increasing distances.

Aligned groups impose fetch restrictions that reduce fetch efficiency.

More stalls due to data dependencies.

More squashes due to branches.

VLIW

Difference with superscalar: instruction bundling.

Deeper Pipelining

Deeper (Super) Pipelining

Relationship between stage splitting, clock frequency and performance.

Relationship between stage splitting and cost.

Latch setup time.

More stalls due to data dependencies.

Vector Instructions

Vector (SIMD, Data Parallel) Instructions

Vector registers.

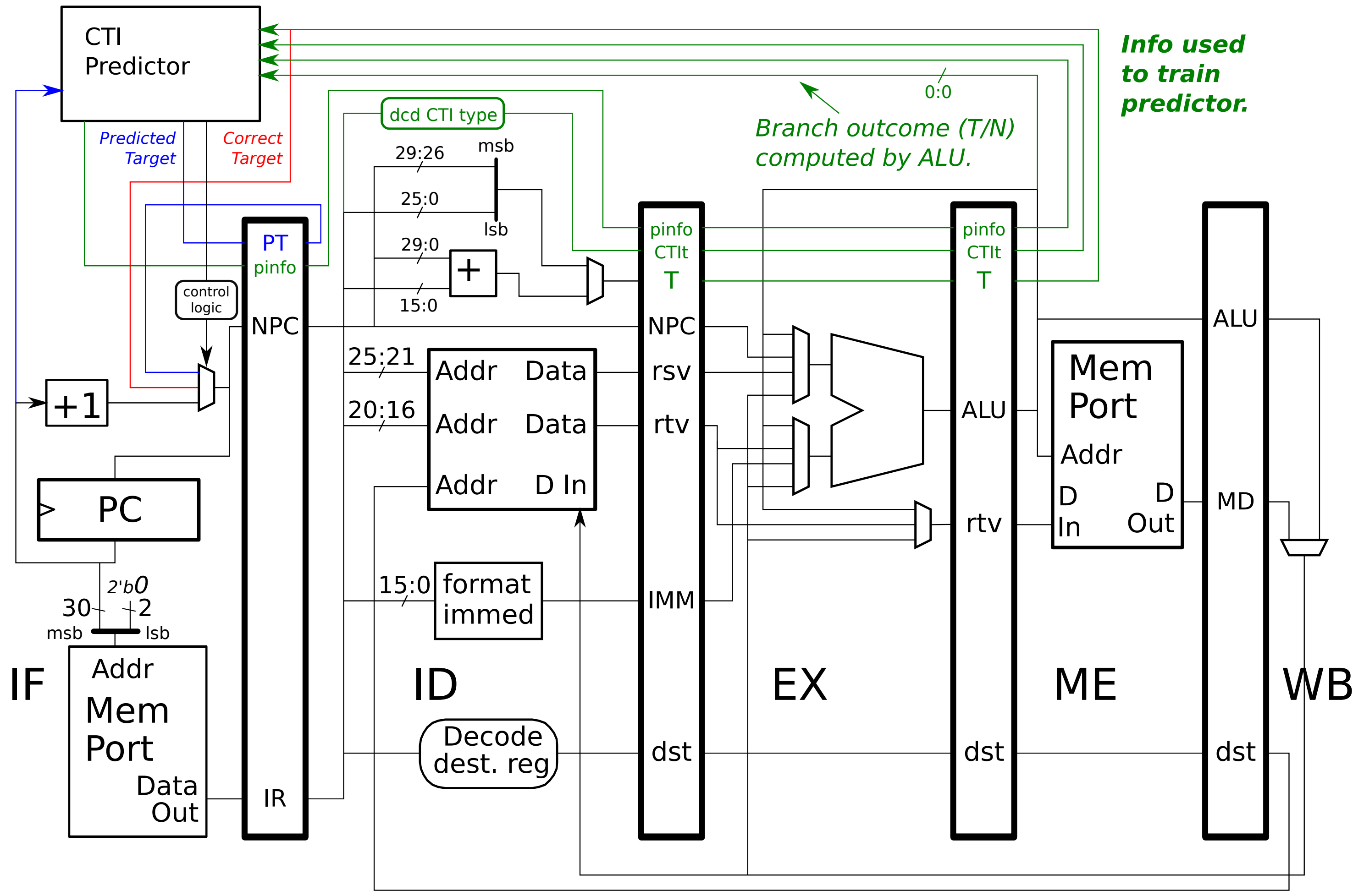
How vector instructions operate on vector registers.

Advantages and disadvantages and differences ...
... between vector instructions and superscalar systems.

CTI Prediction

Definitions, Overview

- Branch Direction Prediction
- CTI Target Prediction
- Where Prediction is Made
- What is Used to Make Prediction
- Misprediction Recovery



Types of Branch Direction Predictors

Bimodal Predictor.

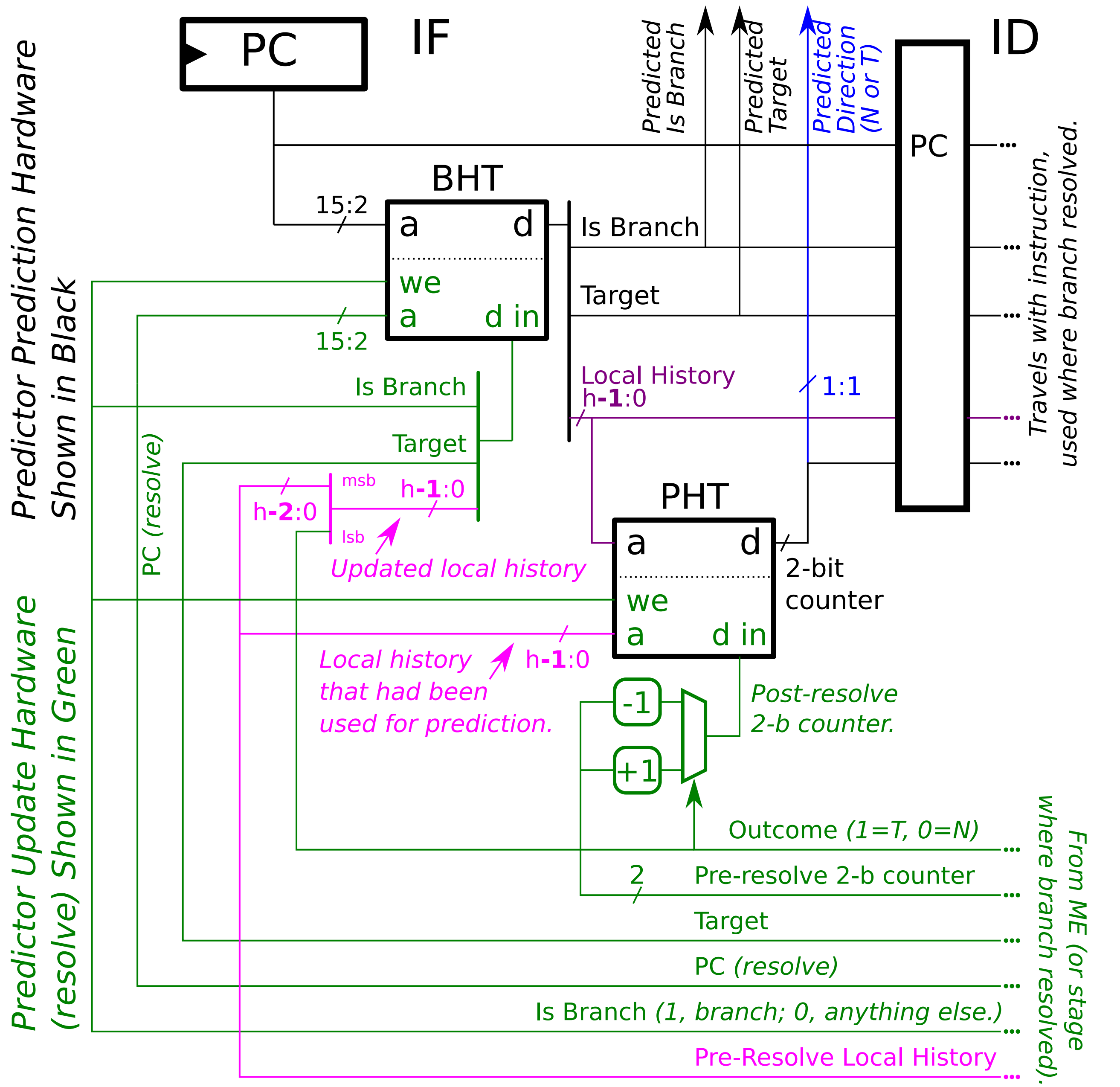
Local History Predictor

Global History Predictor

Global Variations: gshare, gselect.

Less Important Topics

Branch target prediction.



Predictor Prediction Hardware Shown in Black
Predictor Update Hardware (resolve) Shown in Green
PC
IF
ID
BHT
PHT
Predicted Is Branch
Predicted Target
Predicted Direction (N or T)
15:2
a
d
we
a
d in
Is Branch
Target
Local History h-1:0
1:1
PC
Travels with instruction, used where branch resolved.
PC (resolve)
msb
lsb
h-2:0
h-1:0
Updated local history
2-bit counter
Local history that had been used for prediction.
Post-resolve 2-b counter.
-1
+1
Outcome (1=T, 0=N)
2
Pre-resolve 2-b counter
Target
PC (resolve)
Is Branch (1, branch; 0, anything else.)
Pre-Resolve Local History
From ME (or stage where branch resolved).

Caches and Memory

General

Definitions

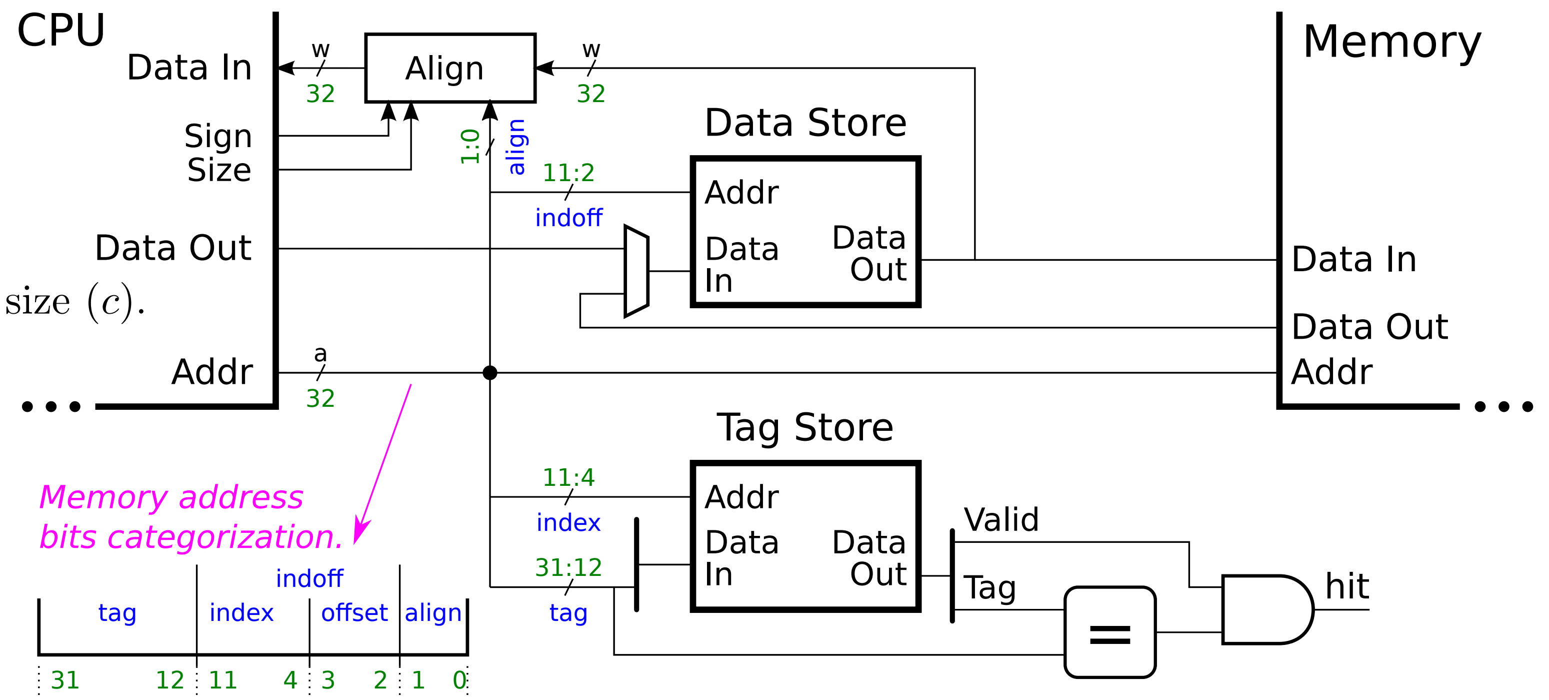
Bus width (w), address space size (a), character size (c).

Caches

Cache structure, connection of memory devices.

Line size implications.

Computing hit ratio for given program.



tag (31–12)	index (11–4)	offset (3–2)	align (1–0)	
0x5	0x43	10_2	01_2	lbu 0x5439
0x1	0x27	00_2	00_2	lw 0x1270
0x5	0x43	10_2	00_2	lbu 0x5438
0x6	0x43	10_2	01_2	lbu 0x6439

Set Associative Caches

Definitions:

line (block), index, tag, alignment, associativity

Connection of tag and data memory.

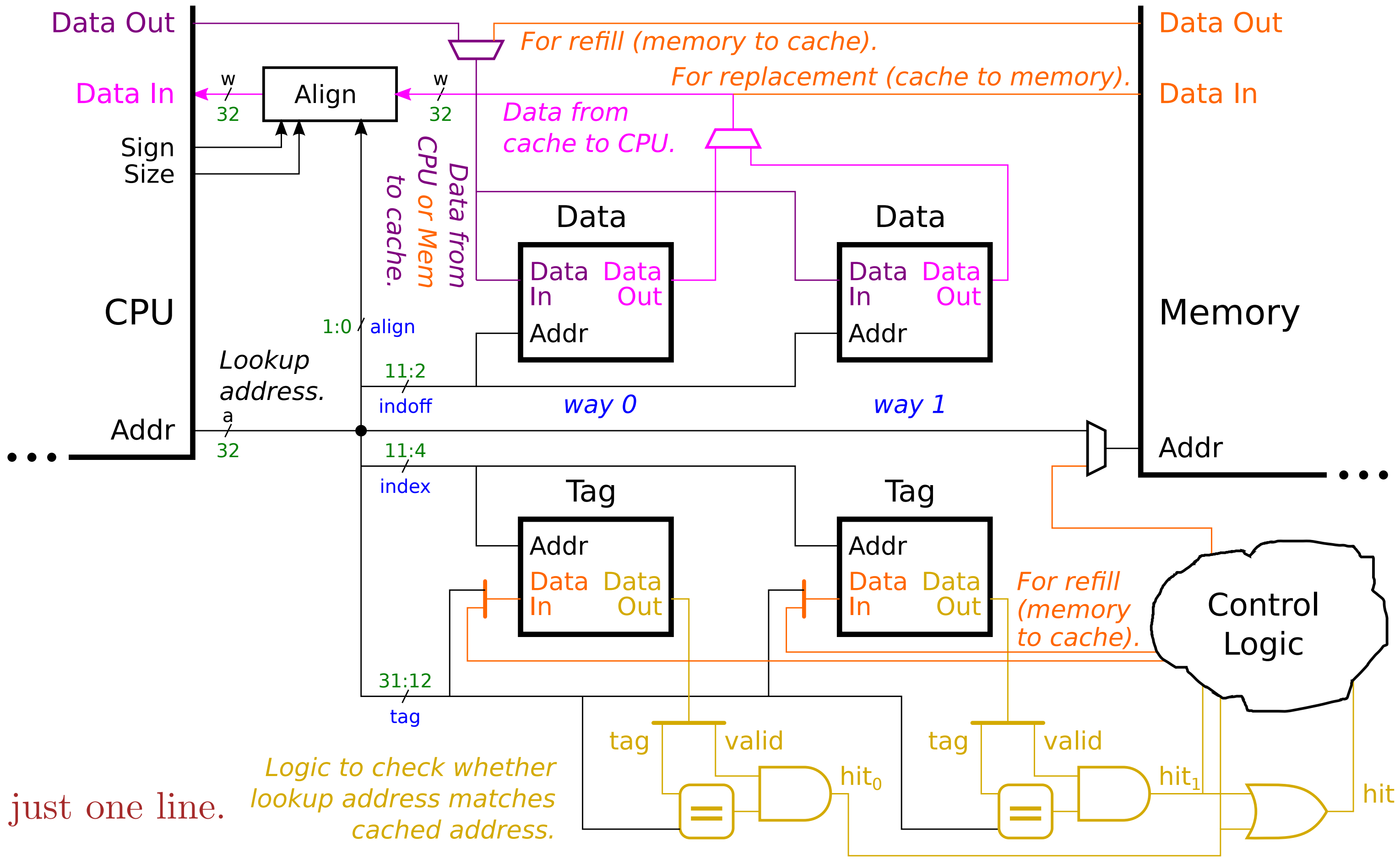
Special Cases:

Direct Mapped:

Not set-associative (1-way).

Fully Associative:

Maximum associativity, each Tag Store describes just one line.


Data Out
Data In
Sign
Size
CPU
Addr
w
32
Align
w
32
For refill (memory to cache).
For replacement (cache to memory).
Data from cache to CPU.
Data from CPU or Mem to cache.
Data Out
Data In
Memory
Addr
Data
Data In
Data Out
Addr
Data
Data In
Data Out
Addr
1:0 align
11:2 indoff
way 0
way 1
Lookup address.
a
32
11:4 index
Tag
Addr
Data In
Data Out
Tag
Addr
Data In
Data Out
For refill (memory to cache).
Control Logic
31:12 tag
tag
valid
tag
valid
hit0
hit1
hit
Logic to check whether lookup address matches cached address.