CSEE 4824 Final Project

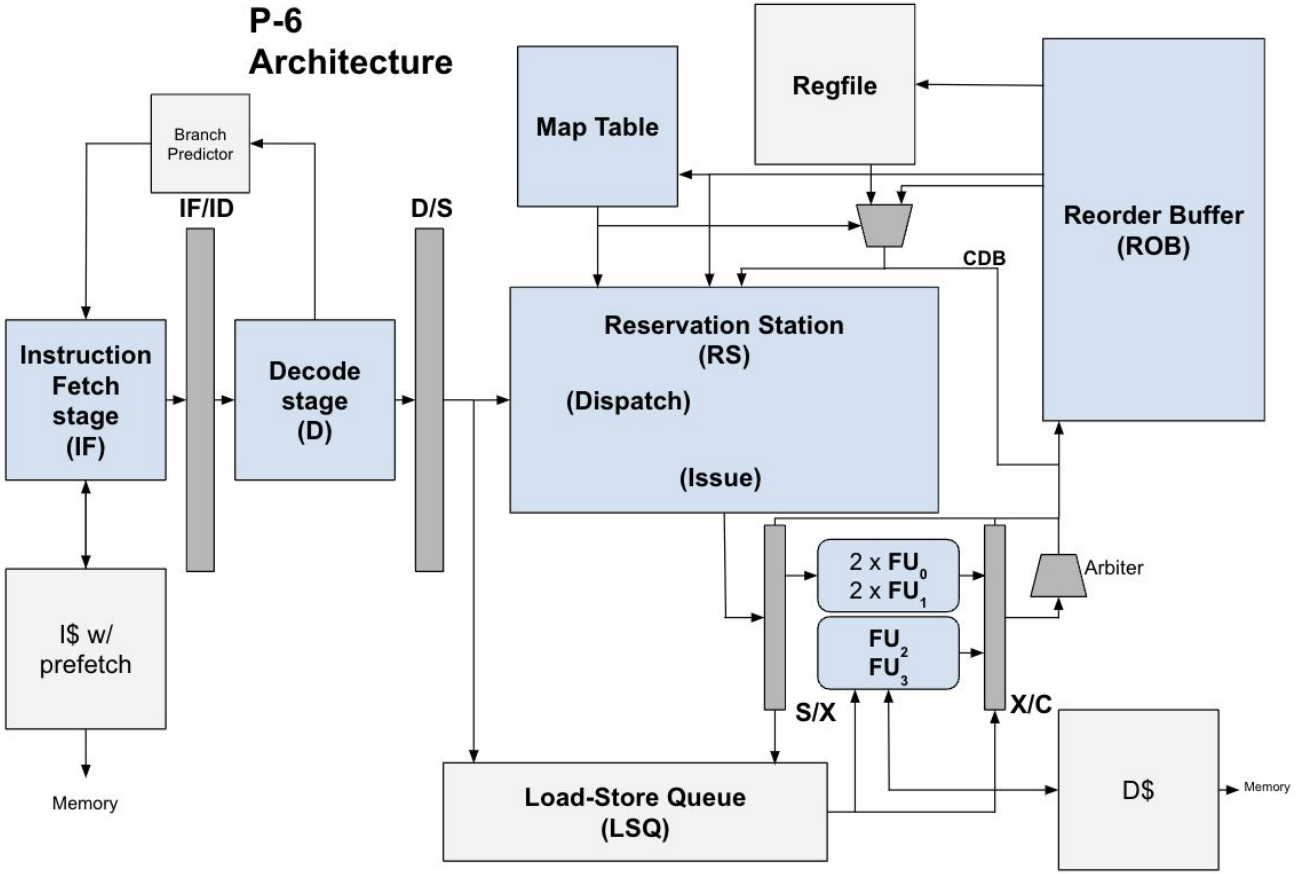
Out-of-Order RISC-V P6 Processor

Group 2
Nicolas Alarcon, Donovan Sproule, Yun-Rong (Alice) Du

Features

- **Simple features**
  Non-blocking L1 data cache
  Non-blocking instruction cache with prefetching
  Load-Store Queue with data forwarding
  Good GUI Debugger

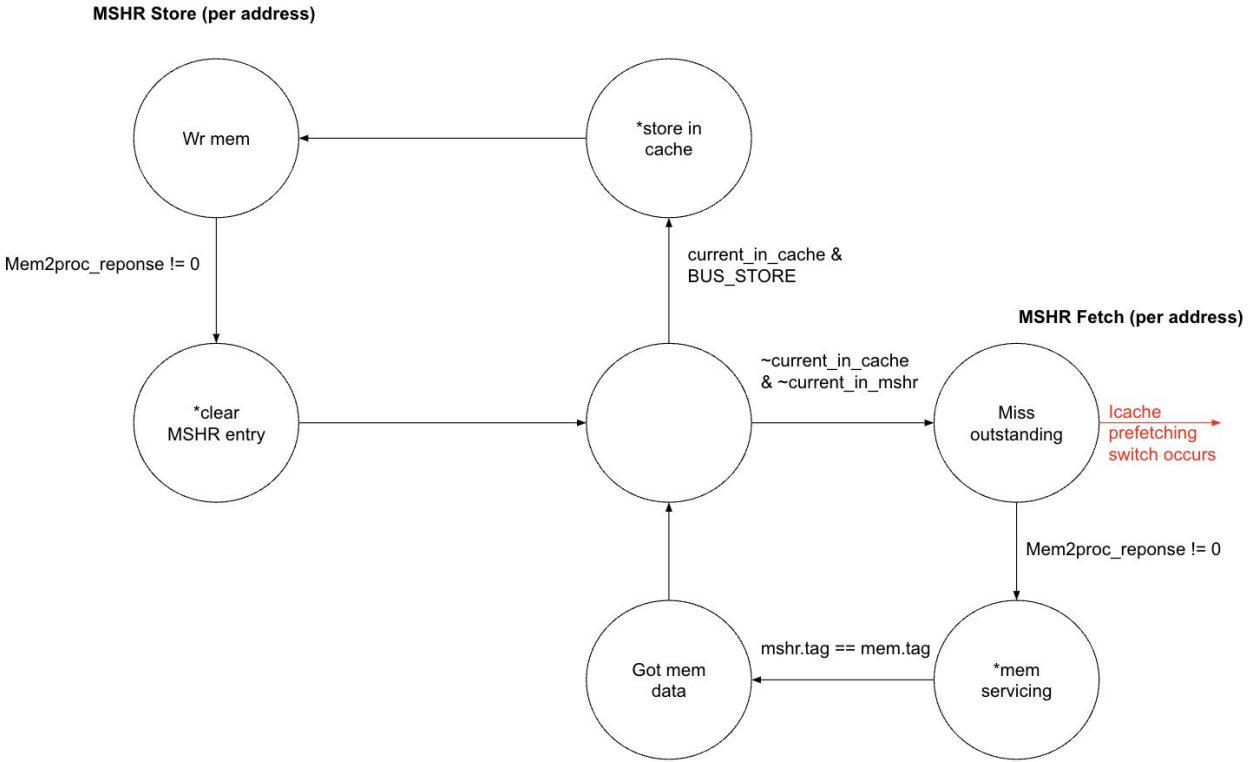
- **Basic features**
  Basic P6 out-of-order processor
  Four functional units: ALU / MUL / LOAD / STORE
  Instruction Cache (size: 256 B) & Data Cache (size: 256 B)
  Two-bit saturating branch predictor


P-6
Architecture
Branch
Predictor
IF/ID
D/S
Map Table
Regfile
Reorder Buffer
(ROB)
CDB
Instruction
Fetch
stage
(IF)
Decode
stage
(D)
Reservation Station
(RS)
(Dispatch)
(Issue)
2 x FU0
2 x FU1
FU2
FU3
Arbiter
I$ w/
prefetch
Memory
S/X
X/C
Load-Store Queue
(LSQ)
D$
Memory

Execution Stage:

- ALU (2)
- MUL (2)
- LOAD (1)
- STORE (1)
- Arbiter (6-bit rotating priority selector)

Non-Blocking MSHR cache system



Prefetching Results/Rationale

Prefetch Offset	Memory Address		
	0	4
+0	0 | 0 | 0	0 | 0 | 0
+4	4 | 0 | 0	8 | 8 | 1
+8	8 | 8 | 1	12 | 8 | 1
+12	12 | 8 | 1	16 | 16 | 2

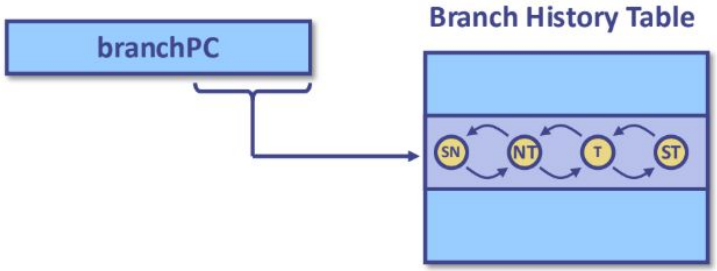
Caching	LSQ	Prefetch	Average CPI
			7.59
X			6.49
X	X		5.54
X	X	X	4.16

Each slot shows:

mem address | aligned mem address | cache line

Branch Predictor

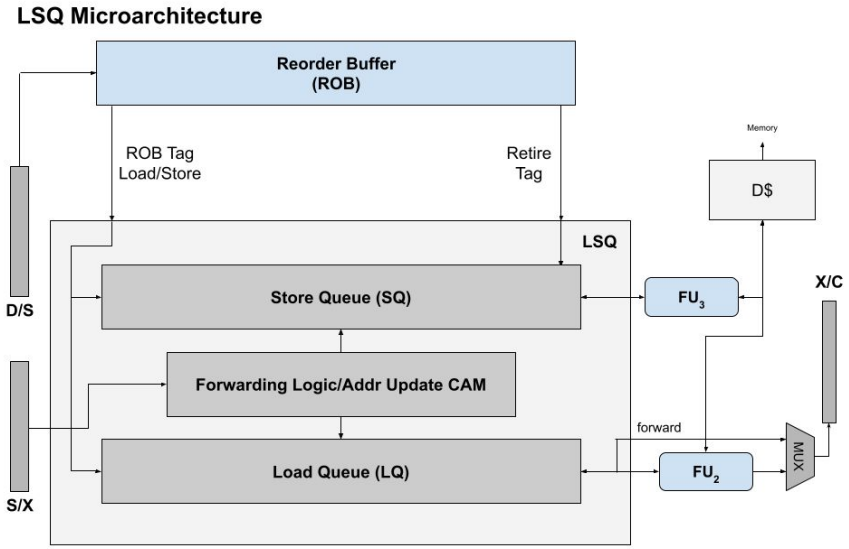
- PC-indexed 2-bit saturating branch predictor
- Save state in bit vector to reduce area



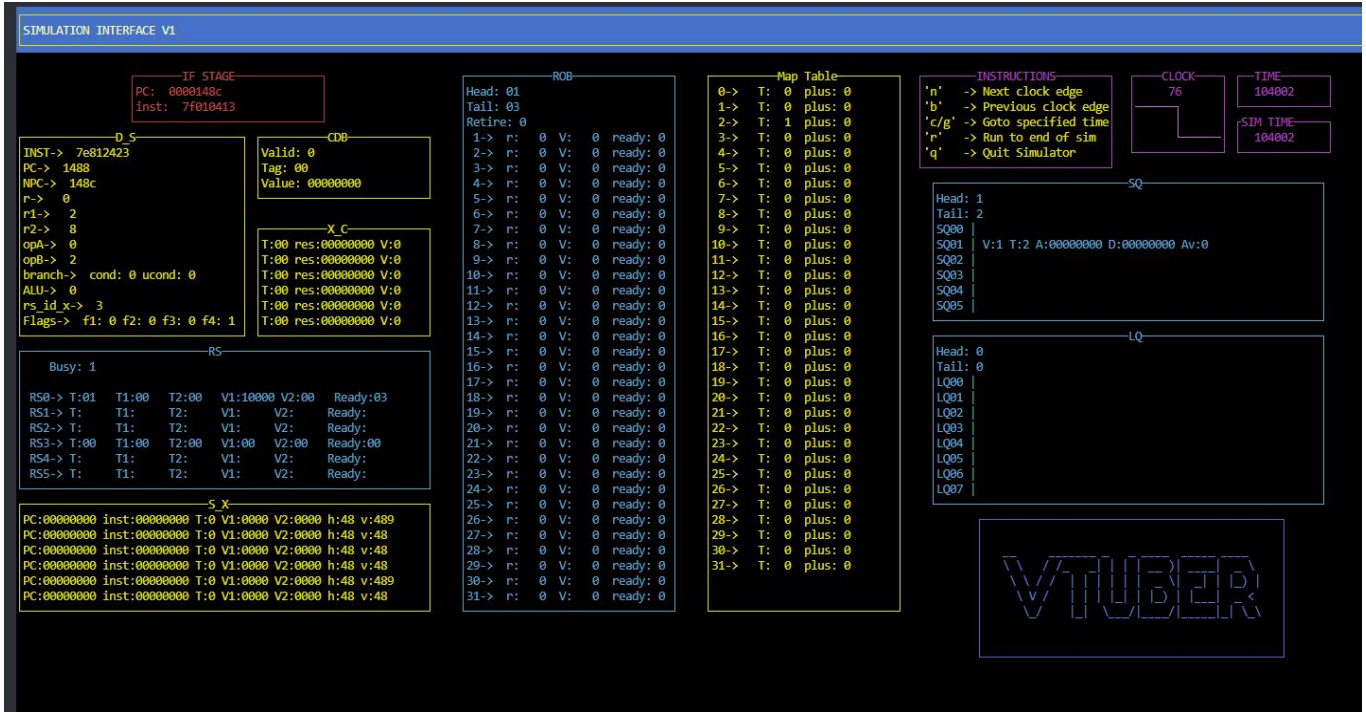
Branch Predictor Type	Average CPI
Always not taken	3.61
Always taken	3.35
BHT	3.31

Memory Operation: Load-Store Queue

- In Order Store/Loads
  - Allocate in Dispatch
    - keep track of dependencies
  - Update Address/Data in Execute ⟶ Forwarding
  - Free if no dependencies
    - Resolve memory request
    - Broadcast if forwarded



Visual Debugger



Summary

Design Specifications:

RS	ROB	AR	LQ	SQ	BHT	MSHR
6 entries	32 entries	32 regs	8 entries (25% of ROB)	6 entries (18.8% of ROB)	256 entries	5 entries

FUs	Cache Assoc.	I- & D-cache	Prefetcher
2 ALUs, 2 Mults 1 Load, 1 Store	1	256 B	1 stride

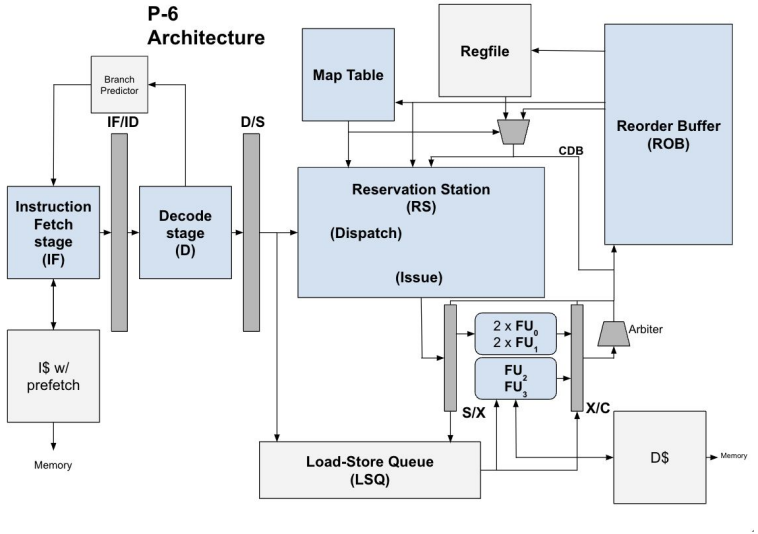
Summary

- Clock Period: 1.3ns
- CPI Performance Improvements for Advanced Features:

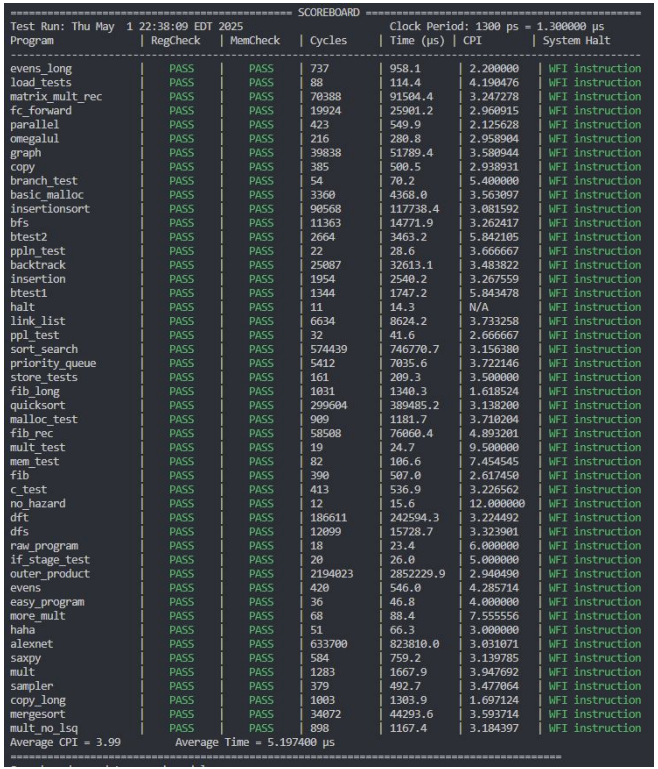
Caching	LSQ	Prefetch	Average CPI
			7.59
X			6.49
X	X		5.54
X	X	X	4.16

Out-of-Order RISC-V P6 Processor (Group #2)

- **Simple features**:
  Non-blocking L1 data cache
  Non-blocking instruction cache with prefetching
  Load-Store Queue with data forwarding
  Good GUI Debugger
- **Basic features**:
  Basic Synthesizable P6 out-of-order processor
  Four functional units: ALU / MUL / LOAD / STORE
  Instruction Cache & Data Cache
    - Size of both: 256 B
  Two-bit saturating branch predictor
- Average CPI is **3.99** and clock period is **1.3ns, 769.2MHz frequency**



Summary



SCOREBOARD

Test Run: Thu May 1 22:38:09 EDT 2025 Clock Period: 1300 ps = 1.300000 µs

Program	RegCheck	MemCheck	Cycles	Time (µs)	CPI	System Halt
evens_long	PASS	PASS	737	958.1	2.200000	WFI instruction
load_tests	PASS	PASS	88	114.4	4.190476	WFI instruction
matrix_mult_rec	PASS	PASS	70388	91504.4	3.247278	WFI instruction
fc_forward	PASS	PASS	19924	25901.2	2.960915	WFI instruction
parallel	PASS	PASS	423	549.9	2.125628	WFI instruction
omegalul	PASS	PASS	216	280.8	2.958904	WFI instruction
graph	PASS	PASS	39838	51789.4	3.580944	WFI instruction
copy	PASS	PASS	385	500.5	2.938931	WFI instruction
branch_test	PASS	PASS	54	70.2	5.400000	WFI instruction
basic_malloc	PASS	PASS	3360	4368.0	3.563097	WFI instruction
insertionsort	PASS	PASS	90568	117738.4	3.881592	WFI instruction
bfs	PASS	PASS	11363	14771.9	3.262417	WFI instruction
btest2	PASS	PASS	2664	3463.2	5.842105	WFI instruction
ppln_test	PASS	PASS	22	28.6	3.666667	WFI instruction
backtrack	PASS	PASS	25087	32613.1	3.483822	WFI instruction
insertion	PASS	PASS	1954	2540.2	3.267559	WFI instruction
btest1	PASS	PASS	1344	1747.2	5.843478	WFI instruction
halt	PASS	PASS	11	14.3	N/A	WFI instruction
link_list	PASS	PASS	6634	8624.2	3.733258	WFI instruction
ppl_test	PASS	PASS	32	41.6	2.666667	WFI instruction
sort_search	PASS	PASS	574439	746770.7	3.156380	WFI instruction
priority_queue	PASS	PASS	5412	7035.6	3.722146	WFI instruction
store_tests	PASS	PASS	161	209.3	3.500000	WFI instruction
fib_long	PASS	PASS	1031	1340.3	1.618524	WFI instruction
quicksort	PASS	PASS	299604	389485.2	3.138200	WFI instruction
malloc_test	PASS	PASS	909	1181.7	3.710204	WFI instruction
fib_rec	PASS	PASS	58508	76060.4	4.893201	WFI instruction
mult_test	PASS	PASS	19	24.7	9.500000	WFI instruction
mem_test	PASS	PASS	82	106.6	7.454545	WFI instruction
fib	PASS	PASS	390	507.0	2.617450	WFI instruction
c_test	PASS	PASS	413	536.9	3.226562	WFI instruction
no_hazard	PASS	PASS	12	15.6	12.000000	WFI instruction
dft	PASS	PASS	186611	242594.3	3.224492	WFI instruction
dfs	PASS	PASS	12099	15728.7	3.323901	WFI instruction
raw_program	PASS	PASS	18	23.4	6.000000	WFI instruction
if_stage_test	PASS	PASS	20	26.0	5.000000	WFI instruction
outer_product	PASS	PASS	2194023	2852229.9	2.940490	WFI instruction
evens	PASS	PASS	420	546.0	4.285714	WFI instruction
easy_program	PASS	PASS	36	46.8	4.000000	WFI instruction
more_mult	PASS	PASS	68	88.4	7.555556	WFI instruction
haha	PASS	PASS	51	66.3	3.000000	WFI instruction
alexnet	PASS	PASS	633700	823810.0	3.031071	WFI instruction
saxpy	PASS	PASS	584	759.2	3.139785	WFI instruction
mult	PASS	PASS	1283	1667.9	3.947692	WFI instruction
sampler	PASS	PASS	379	492.7	3.477064	WFI instruction
copy_long	PASS	PASS	1003	1303.9	1.697124	WFI instruction
mergesort	PASS	PASS	34072	44293.6	3.593714	WFI instruction
mult_no_lsq	PASS	PASS	898	1167.4	3.184397	WFI instruction

Average CPI = 3.99 Average Time = 5.197400 µs

Lessons learned: what we would do next/differently?

- The RS relied on a for-loop to handle all entries at once, next time we should develop a singular entry's logic and used bit-vector notation to build the longer module
  - This ended up causing a more complicated issue in earlier debugging but doesn't affect the final design
- Reusable modules like a FIFO or MSHR fetcher
  - Fifos were in the LSQ, ROB, and considered for other areas
  - The icache & dcache use the same mechanism for fetching
- Learned to fear latch logic
  - Our original RS design passed simulation but due to it relying on fully combinational signals, did not synthesize and then failed all tests. We had to redesign the whole thing
- More documentation in testbenches earlier
  - Towards the end we started adding comments explaining why certain tests we made and expected behavior
  - Better than trying to decipher what the intended behavior was during integration verification
- Pipeline memory requests stronger/Pipeline bottlenecks to increase clock frequency
- Would bank d-cache, advance branch predictor more