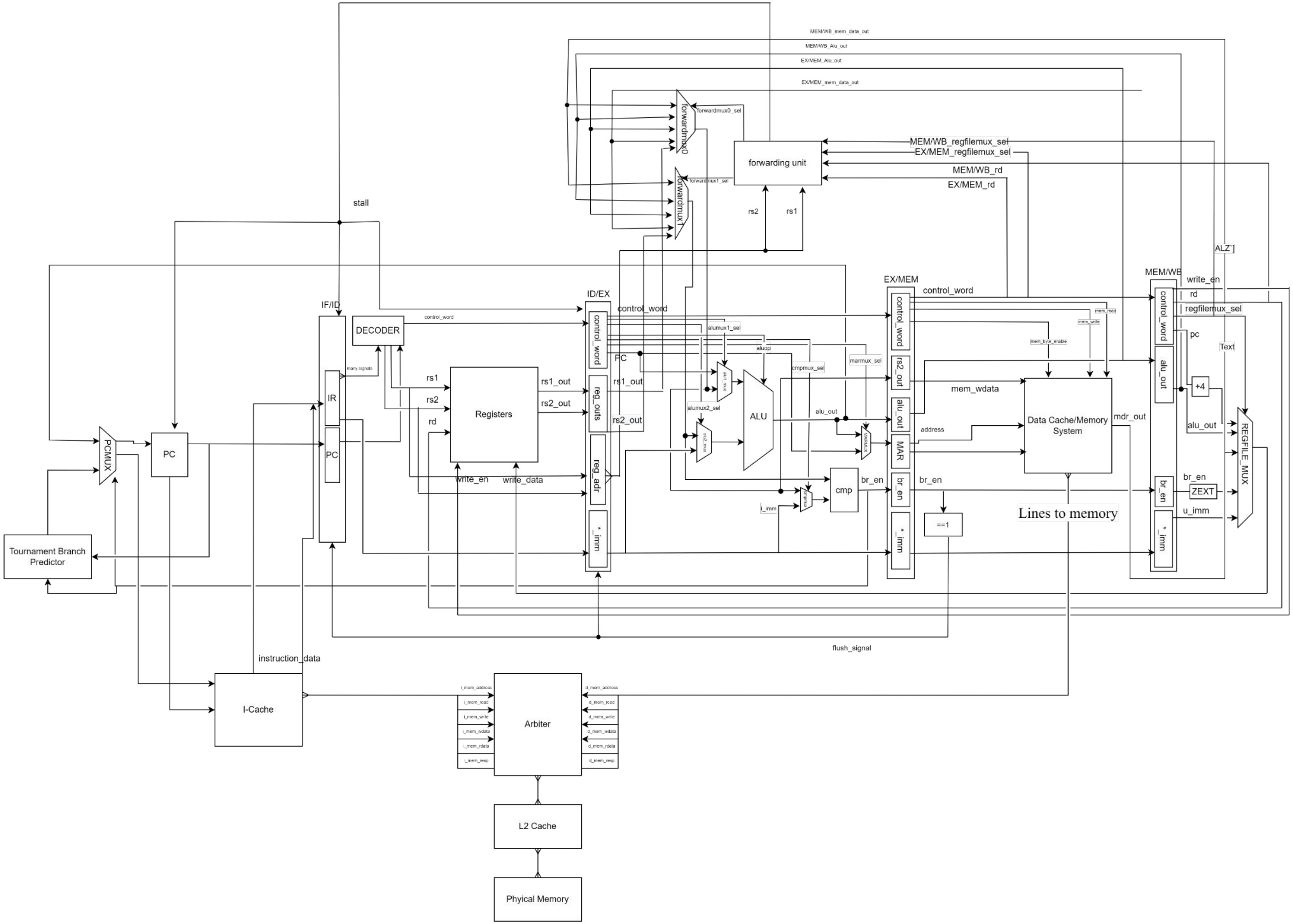
Snapdragon 411

Presented by Daniel Marks, Jason Liu, and Jincheng Liu

Overview

Implemented components

(1) Basic 5-stages pipelined datapath. (cp 1)
(2) Hazard detection & forwarding for the pipelined datapath.(cp2)
(3) L1 Instruction cache , L1 data cache and arbiter. (cp2)
(4) Tournament branch predictor. (cp3)
(5) Pipelined 4-ways cache. (cp3)
(6) L2 cache. (cp3)
(7) Victim cache. (cp3)


MEM/WB_mem_data_out
MEM/WB_Alu_out
EX/MEM_Alu_out
EX/MEM_mem_data_out
forwardmux0
forwardmux0_sel
forwardmux1
forwardmux1_sel
forwarding unit
MEM/WB_regfilemux_sel
EX/MEM_regfilemux_sel
MEM/WB_rd
EX/MEM_rd
rs2
rs1
stall
ALZ']
IF/ID
IR
PC
DECODER
control_word
many signals
rs1
rs2
rd
Registers
rs1_out
rs2_out
write_en
write_data
ID/EX
control_word
reg_outs
reg_adr
*_imm
PC
rs1_out
rs2_out
alumux1_sel
aluop
alumux2_sel
ALU
alu_out
cmpmux_sel
marmux_sel
i_imm
cmp
br_en
EX/MEM
control_word
rs2_out
alu_out
MAR
br_en
*_imm
==1
mem_wdata
address
mem_byte_enable
mem_write
mem_read
Data Cache/Memory System
mdr_out
Lines to memory
MEM/WB
write_en
rd
regfilemux_sel
pc
Text
control_word
alu_out
+4
alu_out
br_en
br_en
ZEXT
u_imm
*_imm
REGFILE_MUX
PCMUX
PC
Tournament Branch Predictor
flush_signal
instruction_data
I-Cache
Arbiter
i_mem_address
i_mem_read
i_mem_write
i_mem_wdata
i_mem_rdata
i_mem_resp
d_mem_address
d_mem_read
d_mem_write
d_mem_wdata
d_mem_rdata
d_mem_resp
L2 Cache
Phyical Memory

Tournament branch predictor

Overview:

- Uses two branch history tables
  - Local branch history table (BTB with 6 bits from PC)
  - Global branch history table (BTB with plus a bit sequence which tracks the outcome of the last 4 branches
- Decides outcome with 2-bit selector which switches between the local and global predictor
  - Similar to the 2-bit branch predictor described in lecture

Challenges:

- Checking for flushing condition
  - Both the prediction and the target address must be correct
- Finding the ideal number of bits to store in the branch prediction table

Tournament branch predictor
Performance Table

	Local: 5 PC bits Global: 4 PC bits, 3 past branch bits	Local: 5 PC bits Global: 4 PC bits, 4 past branch bits	Local: 7 PC bits Global: 5 PC bits, 4 past branch bits
comp1	8860/12528 (70%)	9005/12586 (72%)	10028/12611(80%)
comp2	19802/35414 (56%)	20472/35447 (58%)	25496/35472 (72%)
comp3	3076/4787 (64%)	3172/4754 (67%)	3818/4787 (80%)

Conclusion/Comment: Increasing the number of past prediction bits can help performance, but increasing the number of PC bits is much more effective

Pipelined Cache

Overview:

Using Bram module to achieve next cycle hit, bram enable larger L1 cache with little cost in power and frequencies. Bram module doesn't use FPGA register. When the first tag_out match with mem_address, it is already loading the next tag. So we can do next cycle hit without needing to stall the pipelined CPU.

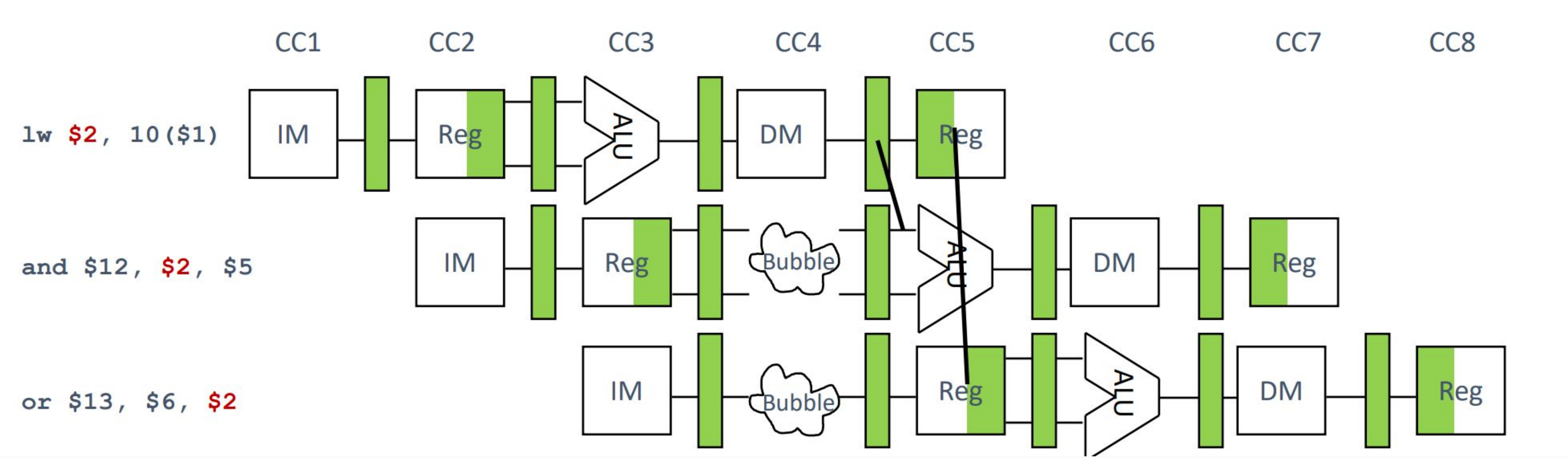
Revising the stage design for l1 icache and dcache.

Changing forwarding path since now we get the data one cycle late or we need to fetch data one cycle early.

Challenges:

- Bram tutorial was given to be very limited. There were a couple of settings that we needed to find out ourselves. The change in forwarding design is needed in both Icache and dcache, namely need to change where instruction data come in and where dcache data come out.
- Implementing bubble states to prevent a data hazard between a load function and a consecutive dependency
- Writing back to memory on dirty evictions

Bubble State Example



Pipelined Cache Performance Table

	DATA CACHE		INSTRUCTION CACHE
	32 Sets, 2 Ways	32 Sets, 4 Ways	32 Sets, 2 Ways
comp1	Misses: 7/3202 (0.21%) Write Backs: 0 Bubbles: 0	Misses: 7/3202 (0.21%) Write Backs: 0 Bubbles: 0	Misses: 25/49431 (.051%)
comp2	Misses: 24/4581 (0.52%) Write Backs: 0 Bubbles: 155	Misses: 24/4581 (0.52%) Write Backs: 0 Bubbles: 155	Misses: 41/108598 (0.038%)
comp3	Misses: 284/11494 (2.5%) Write Backs: 1 Bubbles: 4345	Misses: 283/11374 (2.4%) Write Backs: 1 Bubbles: 4345	Misses: 30/63515 (0.047%)

Conclusion/Comment: For the competition programs, the 4 way D-cache did not significantly improve performance!

L2 cache

Overview:

- We could obtain a L2 cache by removing the bus adaptor of a L1 cache
- The L2 sits in between the arbiter and the physical memory

Challenges:

- Being careful when connecting the ports
- Transferring data between L1 and L2
- Choosing a good size for the L2

L2 Cache
Performance Table

	DATA CACHE	
	L2	L1
comp1	Misses: 20/20 Write Backs: 0	Misses: 1/209 (0.21%) Write Backs: 1 Bubbles: 0
comp2	Misses: 20/20 Write Backs: 0	Misses: 24/4581 (0.52%) Write Backs: 0 Bubbles: 155
comp3	Misses: 304/304 Write Backs: 276	Misses: 283/11374 (2.4%) Write Backs: 1 Bubbles: 4345

Conclusion/Comment:

Victim Cache

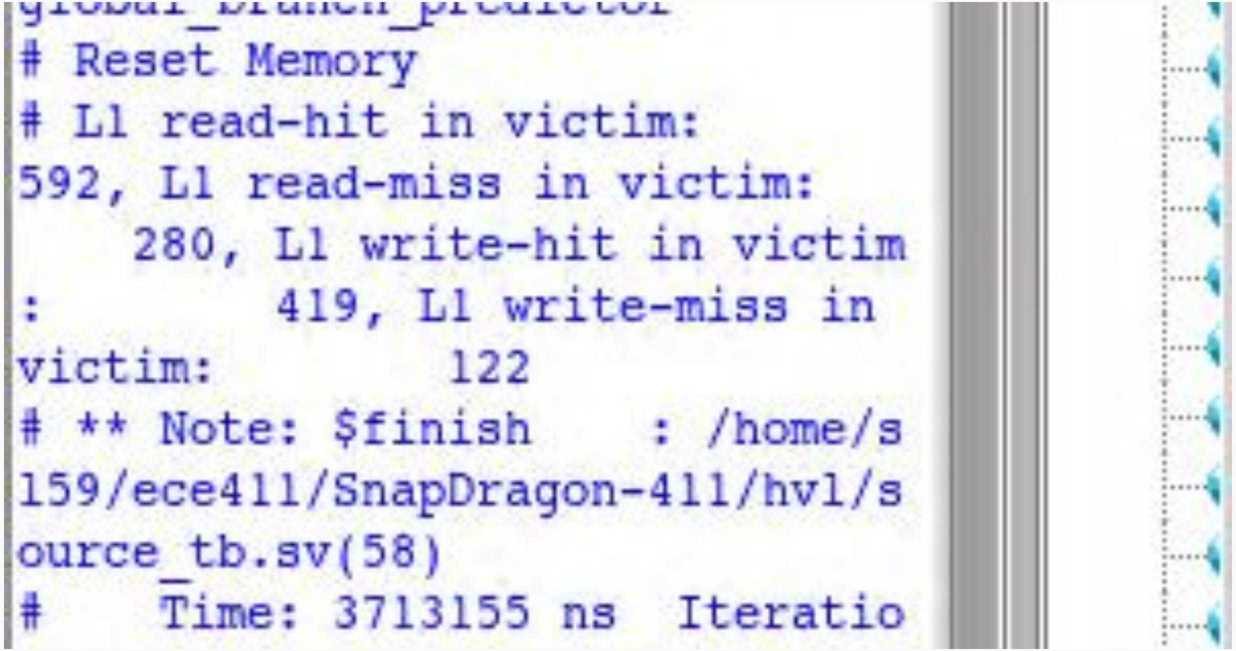
Overview:

- Create several arrays with length 16 to store valid bit, dirty bit, address(as key), and data.
- Implement lru queue to keep track of lru data.
- Only comes to victim cache on upper cache misses.
- Evicted data whether clean or dirty will be put into victim cache. Victim cache only ever write back when its own evicted data are deemed dirty.
- Icache doesn't do any write back so we only consider dcache/l2cache case.

Challenge: We tried to create one separately so it was first tested on given cache. Given cache was one cycle hit and it ends up creating a lot more trouble than we anticipated, it was extremely difficult to debug using modelsim because every error was created way earlier in the process by wrongly write/writeback data.

Duplicate entry in the victim cache has to be rearrange properly or write back

Victim Cache
Performance on given dcache



Total misses:1413

Total found in victim:1011

Total miss in victim:402

Overall Performance

	Comp1 Time (ns)	Comp1 Power (mW)	Comp1 Score
Baseline	720,755.00	448.44	1.68E-10
Snapdragon	332137.9975	607.89	2.22731011E-11
	Comp2 Time (ns)	**Comp2 Power (mW)**	**Comp2 Score**
Baseline	4,521,285.00	430.48	3.98E-08
Snapdragon	753166.1535	978.48	4.18046256E-10
	Comp3 Time (ns)	**Comp3 Power (mW)**	**Comp3 Score**
Baseline	3,639,265.00	425.27	2.05E-08
Snapdragon	421858.0535	855.98	6.42632449E-11

Geometric Mean =(Comp1Score*Comp2Score*Comp3Score)^(1/3) = 8.42666817×10^−11

Q&A

Thanks!