Project 2: Pipelined CPU + L1 Data Cache

TA: 陳炫均

Announcement

- Individual Project
- Deadline: 6/15 (Tue.) 23:59

- Demo:
  - Time slot: TBD
  - Execute your program before TA and answer a few questions

Specification

Use Verilog to model pipeline CPU with

- Off-chip Data Memory
  - Size: 16KB
  - Data width: 32 Bytes
  - Memory access latency: 10 cycles (send an ack when finish access)
- L1 Data Cache
  - Size: 1KB
  - Associative: 2-way
  - Replacement policy: LRU
  - Cache line size: 32 Bytes
  - Write hit policy: write back
  - Write miss policy: write allocate
  - offset: 5 bits, index: 4 bits, tag: 23 bits

testbench.v

- (optional) Initialize registers in all modules
- Connecting CPU and **off-chip** Data_Memory
- Load `instruction.txt` into instruction memory
- Create clock signal
- Dump Register files & Data memories in each cycle
- Print result to `output.txt` and `cache.txt`

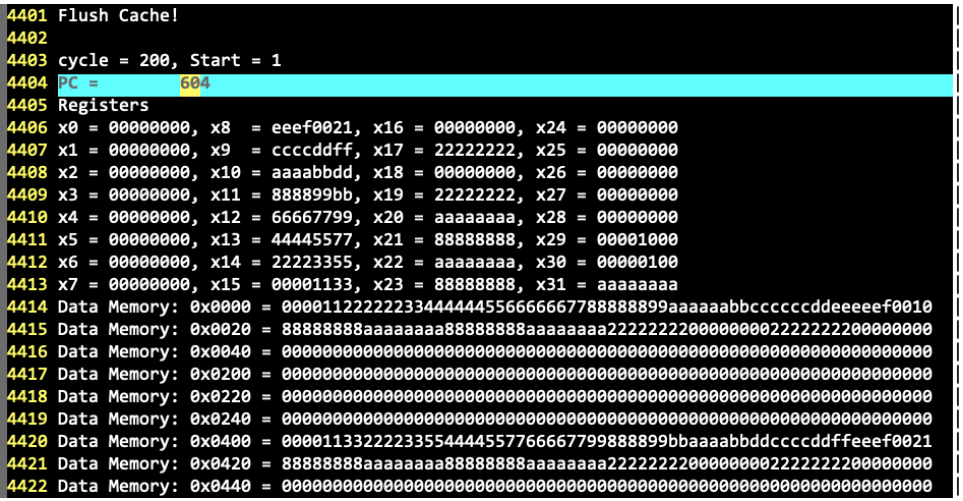
Output Files

- Print result to `output.txt`
  - Output cache status when memory access occurs
  - Criteria: we will check **the final state** is correct or not (The cycle count does not matter)
- Print result to `cache.txt`
  - Record cache hit or cache miss for each cache access
  - Criteria: we will check **the order of hit and miss accesses** is identical to the correct answer (The cycle count does not matter)
- DO NOT CHANGE THE OUTPUT FORMAT

Grading Policy

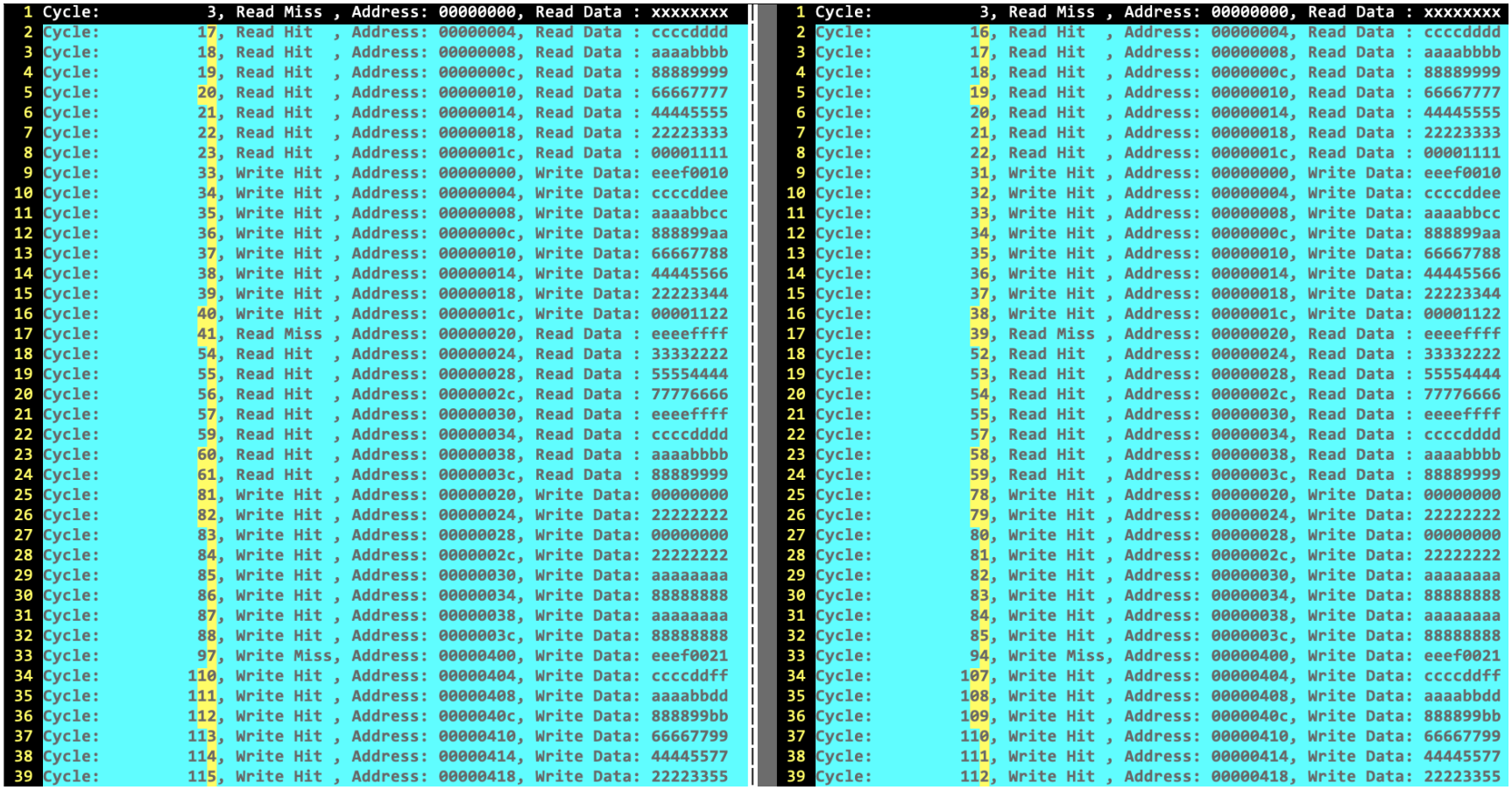
- (80%) Programming
  - You will get 0 point if your code cannot be compiled
  - Grading at demo. You have to answer several questions about how you implement at demo. You may get 0 point on this part if you cannot clearly answer the questions (regarded as plagiarism)
- (20%) Report
  - Implementation of modules
  - Cache controller in detail
    - You can draw a picture to explain if you want
  - Difficulties encountered and solutions of this projects
- Late punishment: 10 points deduction per day

Evaluation Criteria



For output.txt, we will only check the values of registers and data memory at the last cycle. You don’t have to be exactly the same as reference output at every cycle.

Evaluation Criteria (cont.)

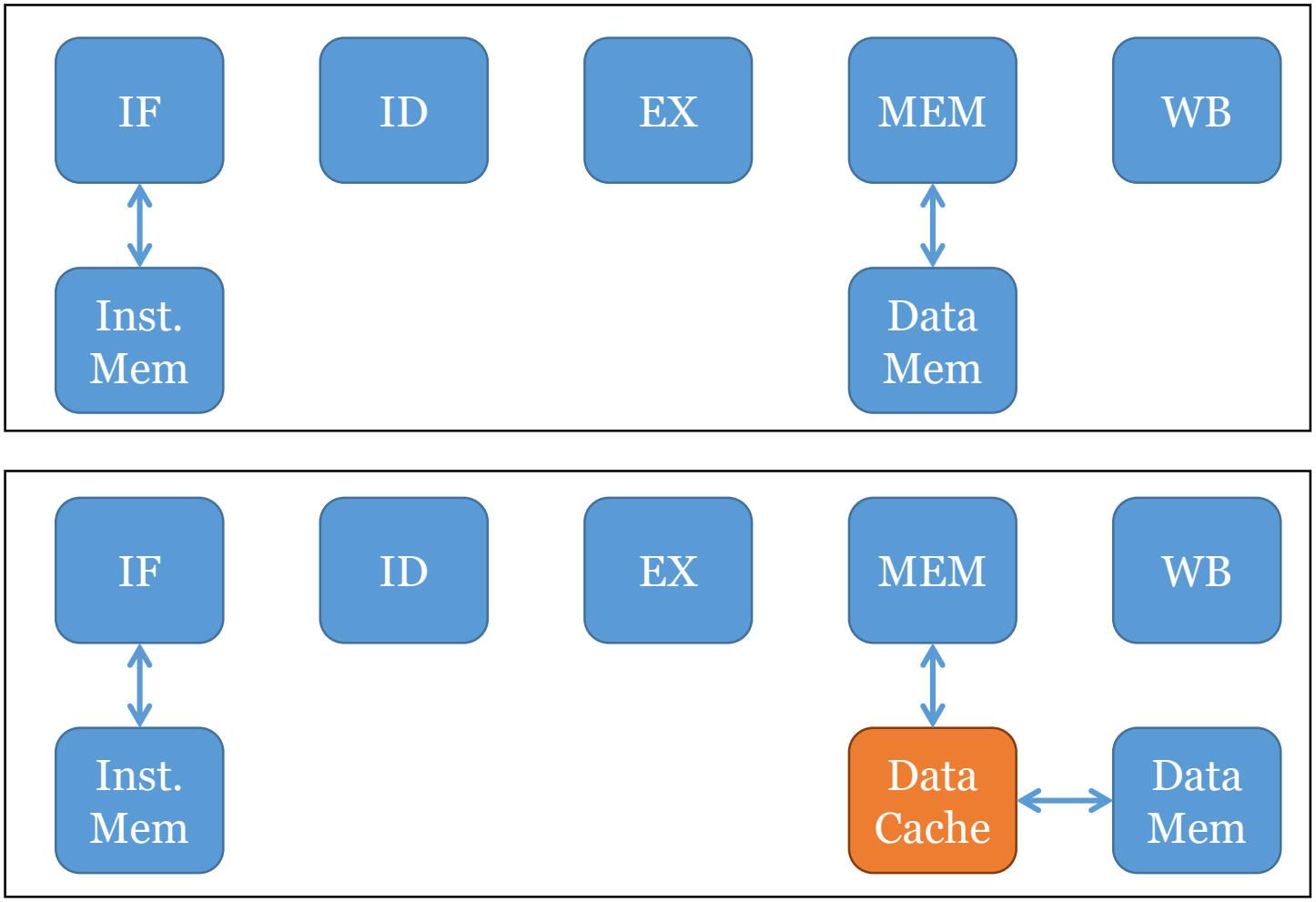


1	Cycle:	3,	Read Miss ,	Address: 00000000,	Read Data :	xxxxxxxx
2	Cycle:	17,	Read Hit ,	Address: 00000004,	Read Data :	ccccdddd
3	Cycle:	18,	Read Hit ,	Address: 00000008,	Read Data :	aaaabbbb
4	Cycle:	19,	Read Hit ,	Address: 0000000c,	Read Data :	88889999
5	Cycle:	20,	Read Hit ,	Address: 00000010,	Read Data :	66667777
6	Cycle:	21,	Read Hit ,	Address: 00000014,	Read Data :	44445555
7	Cycle:	22,	Read Hit ,	Address: 00000018,	Read Data :	22223333
8	Cycle:	23,	Read Hit ,	Address: 0000001c,	Read Data :	00001111
9	Cycle:	33,	Write Hit ,	Address: 00000000,	Write Data:	eeef0010
10	Cycle:	34,	Write Hit ,	Address: 00000004,	Write Data:	ccccddee
11	Cycle:	35,	Write Hit ,	Address: 00000008,	Write Data:	aaaabbcc
12	Cycle:	36,	Write Hit ,	Address: 0000000c,	Write Data:	888899aa
13	Cycle:	37,	Write Hit ,	Address: 00000010,	Write Data:	66667788
14	Cycle:	38,	Write Hit ,	Address: 00000014,	Write Data:	44445566
15	Cycle:	39,	Write Hit ,	Address: 00000018,	Write Data:	22223344
16	Cycle:	40,	Write Hit ,	Address: 0000001c,	Write Data:	00001122
17	Cycle:	41,	Read Miss ,	Address: 00000020,	Read Data :	eeeeffff
18	Cycle:	54,	Read Hit ,	Address: 00000024,	Read Data :	33332222
19	Cycle:	55,	Read Hit ,	Address: 00000028,	Read Data :	55554444
20	Cycle:	56,	Read Hit ,	Address: 0000002c,	Read Data :	77776666
21	Cycle:	57,	Read Hit ,	Address: 00000030,	Read Data :	eeeeffff
22	Cycle:	59,	Read Hit ,	Address: 00000034,	Read Data :	ccccdddd
23	Cycle:	60,	Read Hit ,	Address: 00000038,	Read Data :	aaaabbbb
24	Cycle:	61,	Read Hit ,	Address: 0000003c,	Read Data :	88889999
25	Cycle:	81,	Write Hit ,	Address: 00000020,	Write Data:	00000000
26	Cycle:	82,	Write Hit ,	Address: 00000024,	Write Data:	22222222
27	Cycle:	83,	Write Hit ,	Address: 00000028,	Write Data:	00000000
28	Cycle:	84,	Write Hit ,	Address: 0000002c,	Write Data:	22222222
29	Cycle:	85,	Write Hit ,	Address: 00000030,	Write Data:	aaaaaaaa
30	Cycle:	86,	Write Hit ,	Address: 00000034,	Write Data:	88888888
31	Cycle:	87,	Write Hit ,	Address: 00000038,	Write Data:	aaaaaaaa
32	Cycle:	88,	Write Hit ,	Address: 0000003c,	Write Data:	88888888
33	Cycle:	97,	Write Miss,	Address: 00000400,	Write Data:	eeef0021
34	Cycle:	110,	Write Hit ,	Address: 00000404,	Write Data:	ccccddff
35	Cycle:	111,	Write Hit ,	Address: 00000408,	Write Data:	aaaabbdd
36	Cycle:	112,	Write Hit ,	Address: 0000040c,	Write Data:	888899bb
37	Cycle:	113,	Write Hit ,	Address: 00000410,	Write Data:	66667799
38	Cycle:	114,	Write Hit ,	Address: 00000414,	Write Data:	44445577
39	Cycle:	115,	Write Hit ,	Address: 00000418,	Write Data:	22223355

1	Cycle:	3,	Read Miss ,	Address: 00000000,	Read Data :	xxxxxxxx
2	Cycle:	16,	Read Hit ,	Address: 00000004,	Read Data :	ccccdddd
3	Cycle:	17,	Read Hit ,	Address: 00000008,	Read Data :	aaaabbbb
4	Cycle:	18,	Read Hit ,	Address: 0000000c,	Read Data :	88889999
5	Cycle:	19,	Read Hit ,	Address: 00000010,	Read Data :	66667777
6	Cycle:	20,	Read Hit ,	Address: 00000014,	Read Data :	44445555
7	Cycle:	21,	Read Hit ,	Address: 00000018,	Read Data :	22223333
8	Cycle:	22,	Read Hit ,	Address: 0000001c,	Read Data :	00001111
9	Cycle:	31,	Write Hit ,	Address: 00000000,	Write Data:	eeef0010
10	Cycle:	32,	Write Hit ,	Address: 00000004,	Write Data:	ccccddee
11	Cycle:	33,	Write Hit ,	Address: 00000008,	Write Data:	aaaabbcc
12	Cycle:	34,	Write Hit ,	Address: 0000000c,	Write Data:	888899aa
13	Cycle:	35,	Write Hit ,	Address: 00000010,	Write Data:	66667788
14	Cycle:	36,	Write Hit ,	Address: 00000014,	Write Data:	44445566
15	Cycle:	37,	Write Hit ,	Address: 00000018,	Write Data:	22223344
16	Cycle:	38,	Write Hit ,	Address: 0000001c,	Write Data:	00001122
17	Cycle:	39,	Read Miss ,	Address: 00000020,	Read Data :	eeeeffff
18	Cycle:	52,	Read Hit ,	Address: 00000024,	Read Data :	33332222
19	Cycle:	53,	Read Hit ,	Address: 00000028,	Read Data :	55554444
20	Cycle:	54,	Read Hit ,	Address: 0000002c,	Read Data :	77776666
21	Cycle:	55,	Read Hit ,	Address: 00000030,	Read Data :	eeeeffff
22	Cycle:	57,	Read Hit ,	Address: 00000034,	Read Data :	ccccdddd
23	Cycle:	58,	Read Hit ,	Address: 00000038,	Read Data :	aaaabbbb
24	Cycle:	59,	Read Hit ,	Address: 0000003c,	Read Data :	88889999
25	Cycle:	78,	Write Hit ,	Address: 00000020,	Write Data:	00000000
26	Cycle:	79,	Write Hit ,	Address: 00000024,	Write Data:	22222222
27	Cycle:	80,	Write Hit ,	Address: 00000028,	Write Data:	00000000
28	Cycle:	81,	Write Hit ,	Address: 0000002c,	Write Data:	22222222
29	Cycle:	82,	Write Hit ,	Address: 00000030,	Write Data:	aaaaaaaa
30	Cycle:	83,	Write Hit ,	Address: 00000034,	Write Data:	88888888
31	Cycle:	84,	Write Hit ,	Address: 00000038,	Write Data:	aaaaaaaa
32	Cycle:	85,	Write Hit ,	Address: 0000003c,	Write Data:	88888888
33	Cycle:	94,	Write Miss,	Address: 00000400,	Write Data:	eeef0021
34	Cycle:	107,	Write Hit ,	Address: 00000404,	Write Data:	ccccddff
35	Cycle:	108,	Write Hit ,	Address: 00000408,	Write Data:	aaaabbdd
36	Cycle:	109,	Write Hit ,	Address: 0000040c,	Write Data:	888899bb
37	Cycle:	110,	Write Hit ,	Address: 00000410,	Write Data:	66667799
38	Cycle:	111,	Write Hit ,	Address: 00000414,	Write Data:	44445577
39	Cycle:	112,	Write Hit ,	Address: 00000418,	Write Data:	22223355

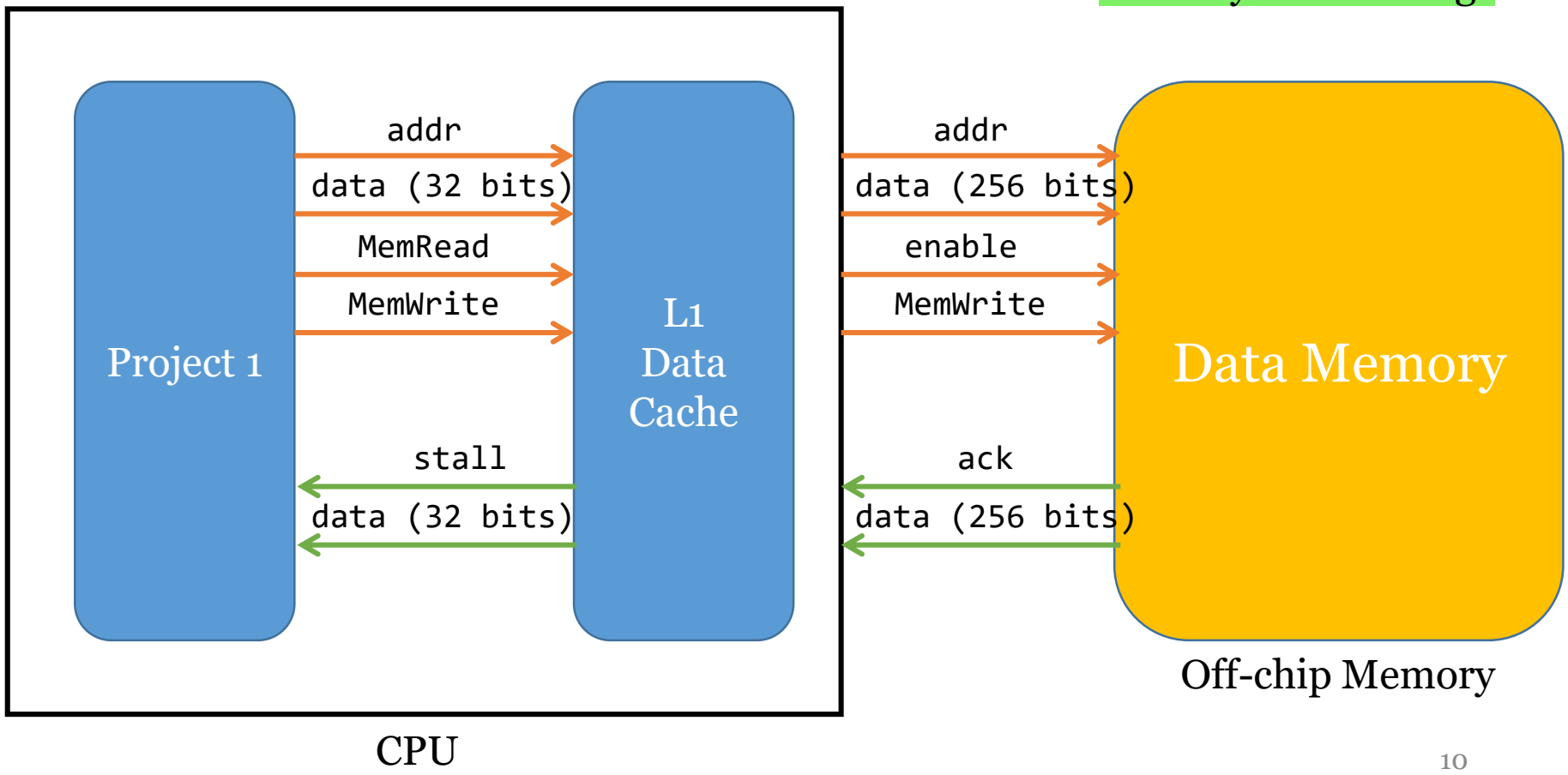
For cache.txt, we will only check number of hit/miss and their order. cycle count doesn't matter.

Project 1 to Project 2

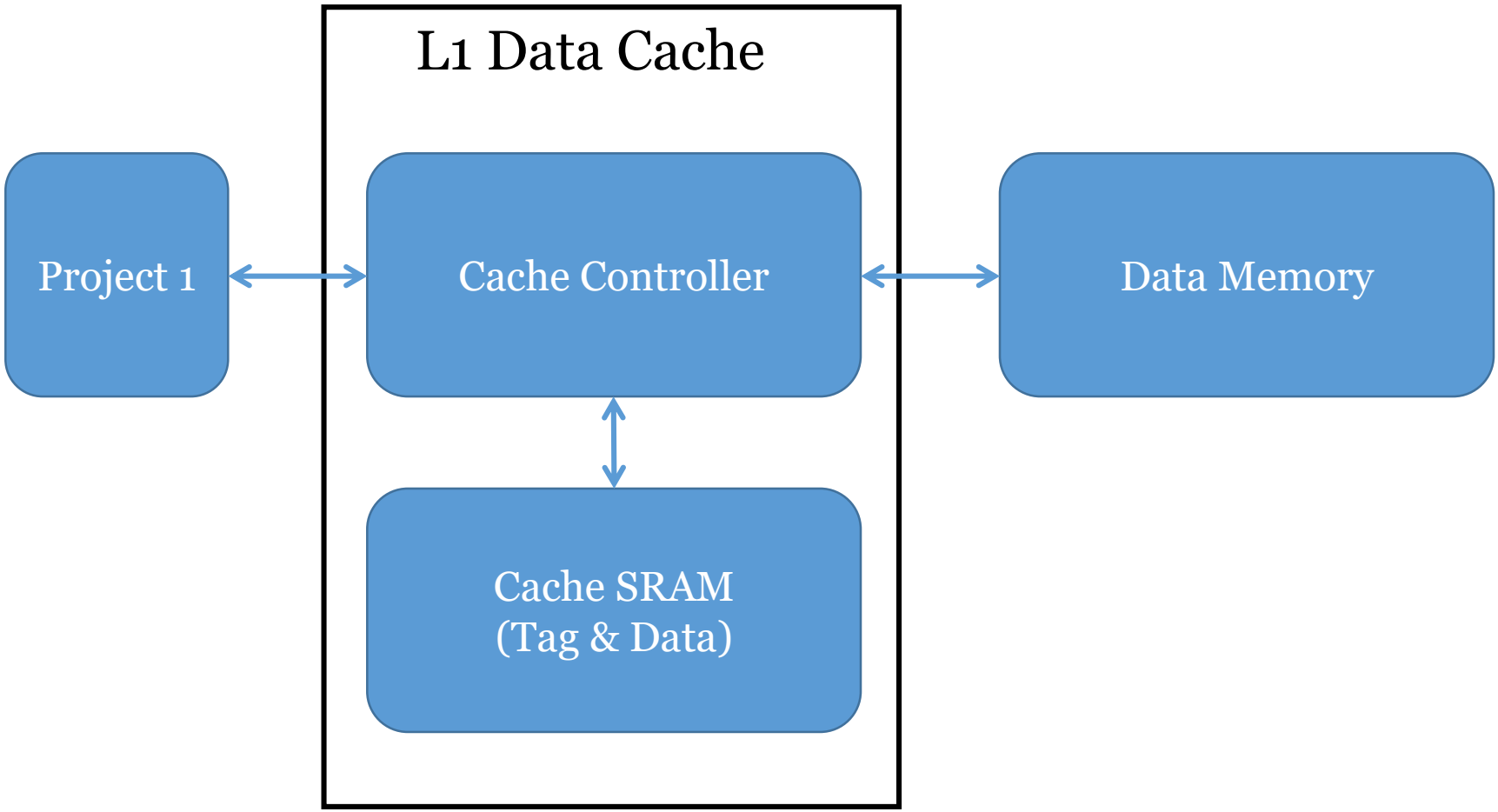


System Block Diagram

enable: memory access enable
write: write data to memory
ack: memory acknowledge



Example files



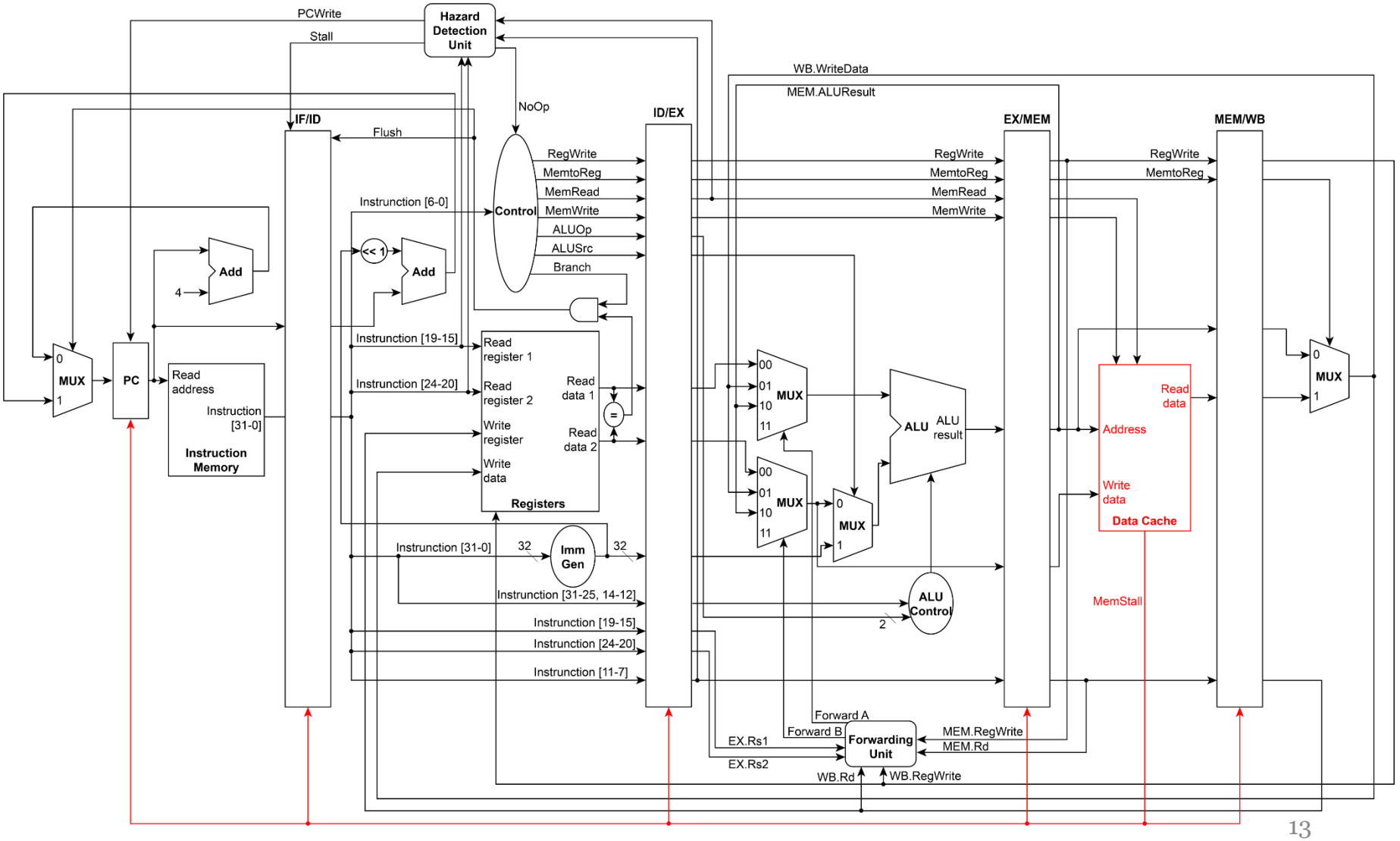
Example files

- `CPU.v`: connection between modules
- `dcache_controller.v`: handles I/O requests
- `dcache_sram.v`: Modify the data structure within it to support 2-way associative cache.
- `Data_Memory.v`
- `testbench.v`

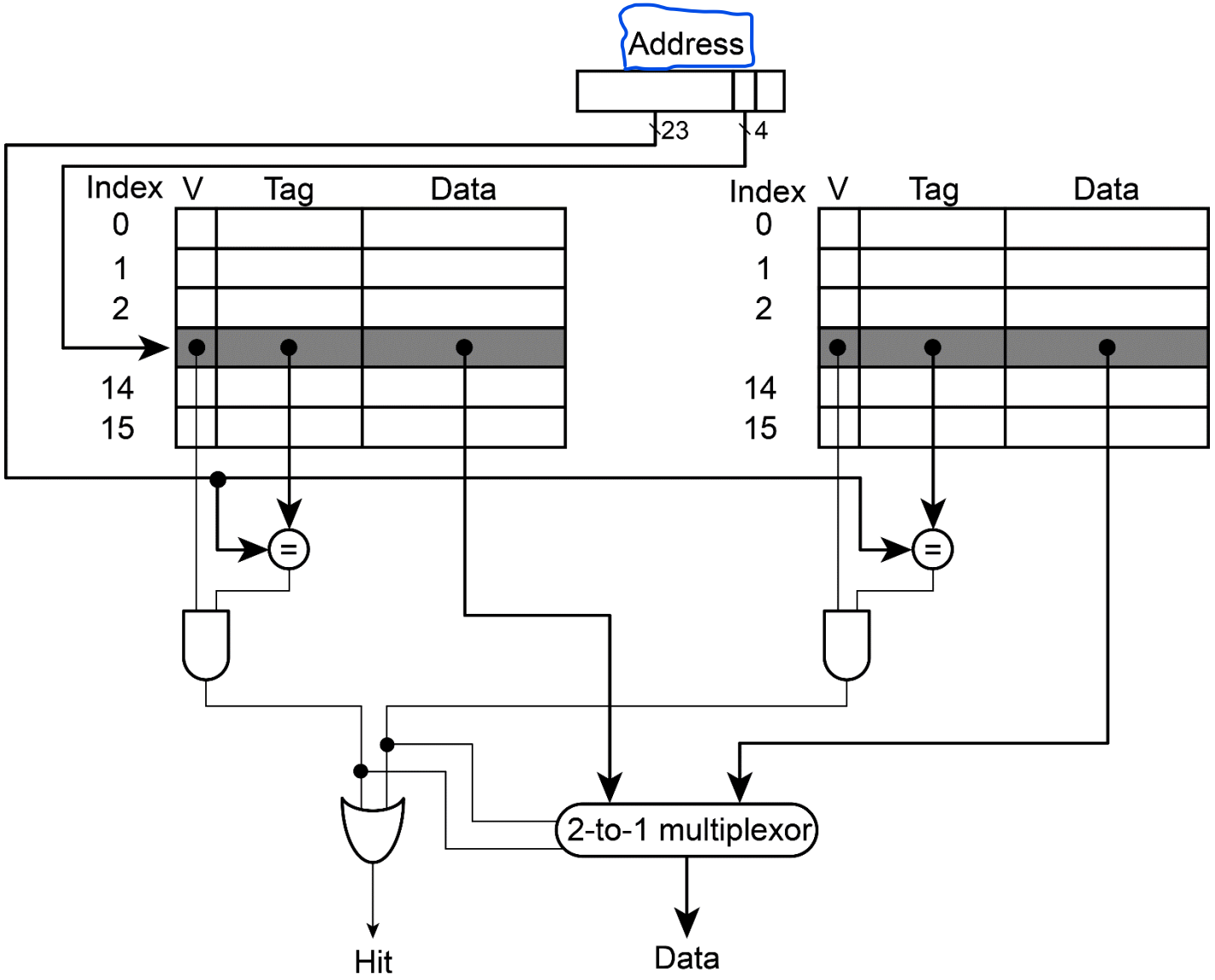
You can modify them as you want
But make sure you include them as submission

- `Instruction_Memory.v`
- `PC.v`
- `Registers.v`

Datapath & Modules



2-way associative cache



Submission Rules

- `studentID_project2 (dir)`
  - `studentID_project2/codes/*.v`
  - `studentID_project2/studentID_project2_report.pdf`
- Pack the above **directory** into a zip file
  - When we unzip your file, the output should be a single directory named studentID_project2

Submission Rules (cont.)

- In testbench.v, you must check the following settings before submission
  - Read instruction from `./instruction.txt`
  - Dump output to `./output.txt` and `./cache.txt`
- Your code can be compiled with the follow command
  - `$ iverilog -o CPU.out *.v`