**HW#02: Memory model, RISC-V Instruction, Decoder**

**Issued:** March 21 (Tue.), 2023 **Due:** March 27 (Mon.), 2023

**What to turn in**: Copy the text from your **MODIFIED** codes and paste it into a document. If a question asks you to plot or display something on the screen, also include the plot and screen output your code generates. Submit either a \*.doc or \*.pdf file.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Problem 1 (15p): Memory model**

Design a memory model in Verilog. Please see the description in the lecture note.

빨간색, 옅은, 레이저이(가) 표시된 사진

자동 생성된 설명

What you have to do:

1. Memory model (15p)

* Design a memory and its test bench based on the baseline codes.
* Submit your RTL files (riscv\_memory.v and riscv\_memory\_tb.v)
* Capture the waveform.

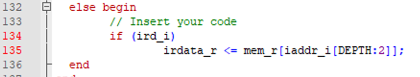
1. Program memory file (5p)

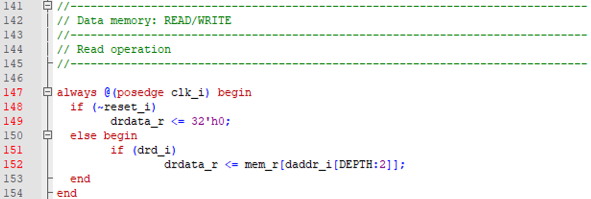
* Use the RISC-V simulator Venus (<https://www.kvakil.me/venus/>) to generate a program file
* Modify the test bench (riscv\_memory\_tb.v) to simulate the memory model with the newly generated file.
* Capture the waveform and the display result printed in Transcript Window.

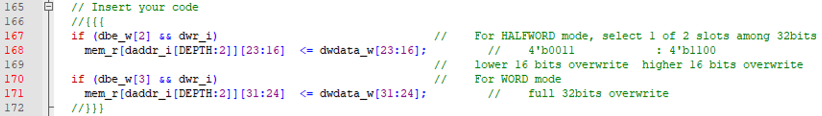
|  |  |  |
| --- | --- | --- |
| **C code** | **Assembly Code** | |
| int A[64];  int sum = 0;  for (int i=0; i<64; i++)  sum += A[i]; | addi x9, x0, 64  add x10, x0, x0  add x11, x0, x0  Loop:    lw x12, 0(x9)    add x10, x10, x12    addi x9, x9, 4    addi x11, x11, 1    addi x13, x0, 64    blt x11, x13, Loop | # x9=&A[0]  # sum=0  # i=0  # x12=A[i]  # sum+= A[i]  # &A[i++]  # i++  # x13=64 |

(a)

- Riscv\_memory.v





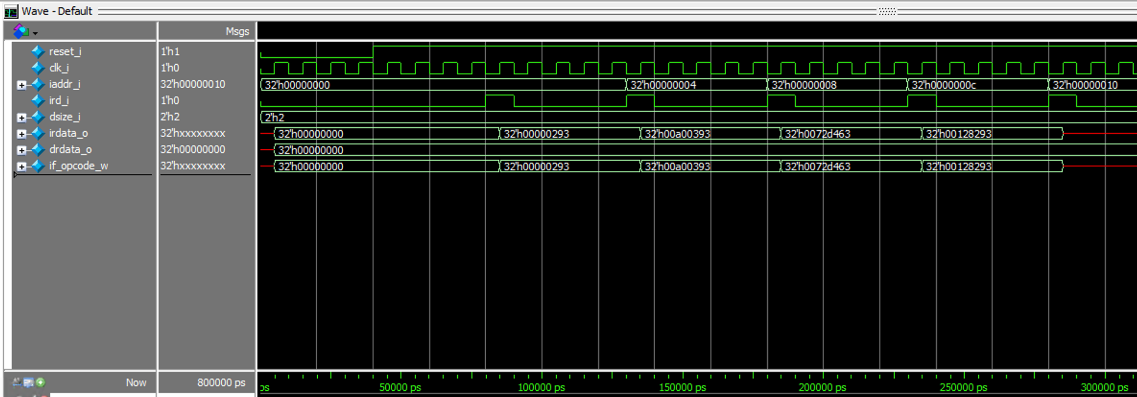


- Riscv\_memory\_tb.v



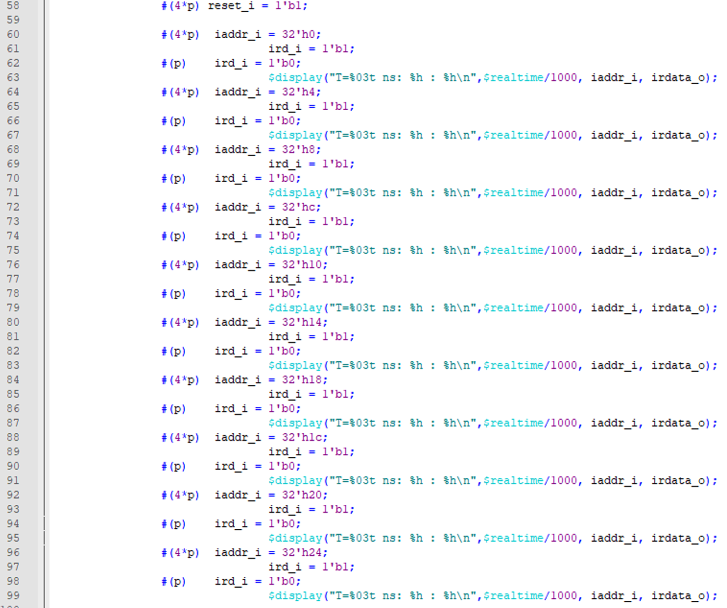
Added if\_opcode\_w wire additionally. ( It was not implemented in the skeleton code. )

- Wave Form

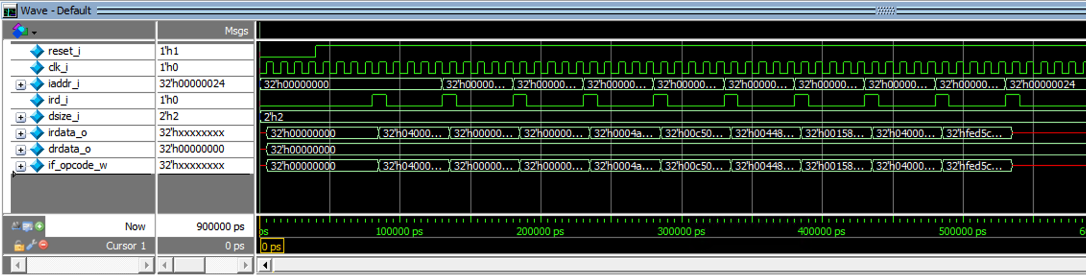


(b)

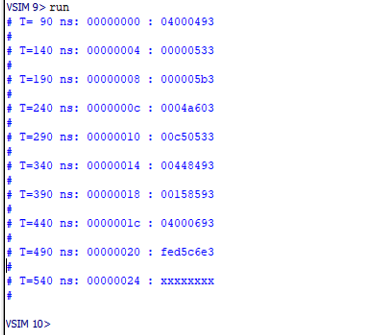
- Riscv\_memory\_tb\_modified.v



- Wave Form

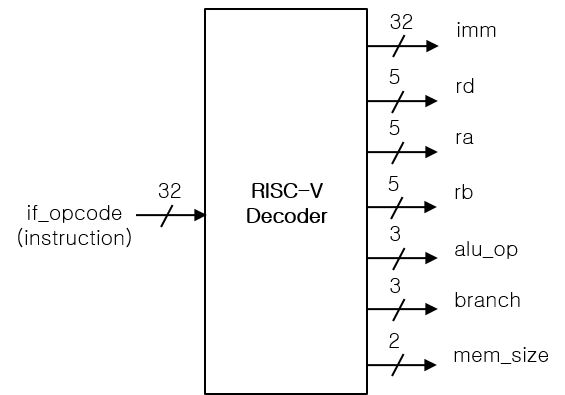


- Transcript Window



**Problem 2 (15p): RISC-V Decoder**

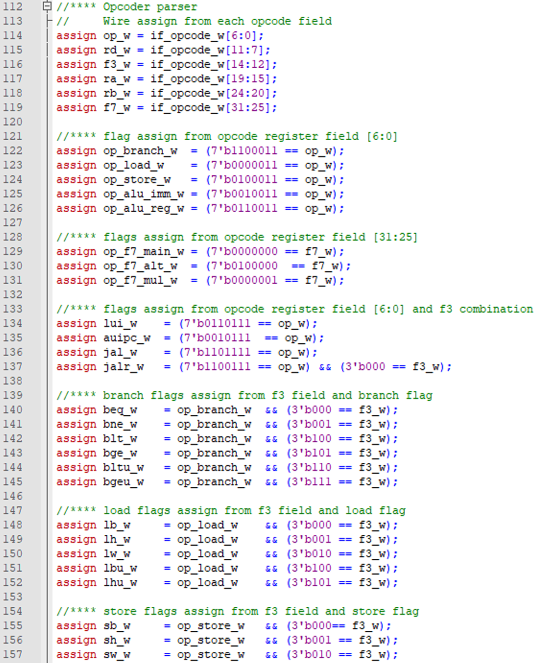
Design a RISC-V decoder in Verilog. Please see the description in the lecture note.

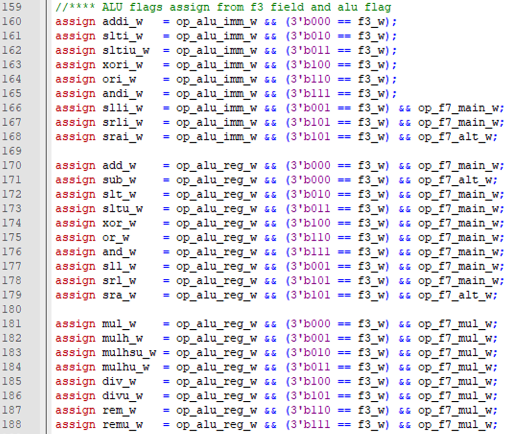


What you have to do:

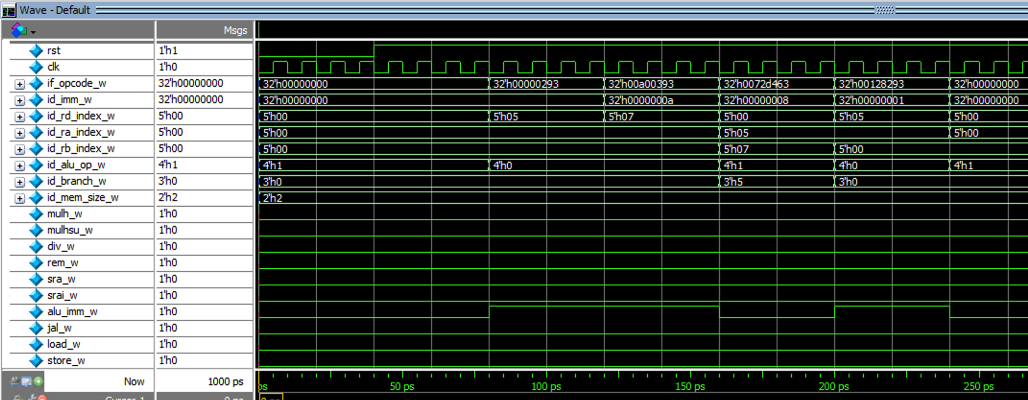
* Design a RISC-V decoder and its test bench based on the baseline codes.
* Submit your RTL files (riscv\_decoder.v and riscv\_decoder\_tb.v).
* Capture the waveform.

- riscv\_decoder.v



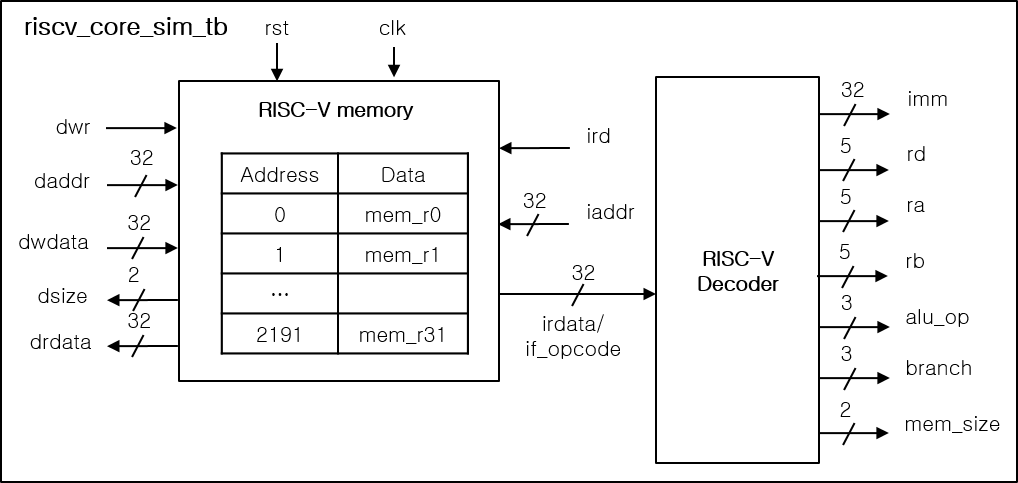


- Wave Form



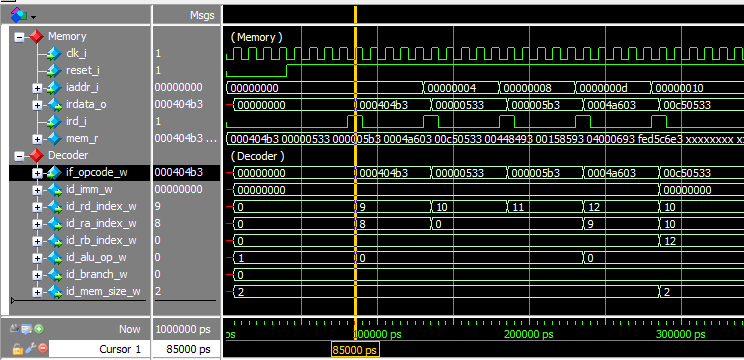
**Problem 3 (5p): Memory & Decoder**

* Complete the test bench riscv\_core\_sim\_tb.v using a memory model in Problem 1 and a decoder in Problem 2.
* Modify the test bench to simulate the newly generated file in **Problem 1b**.
* Capture the waveform.



Example:

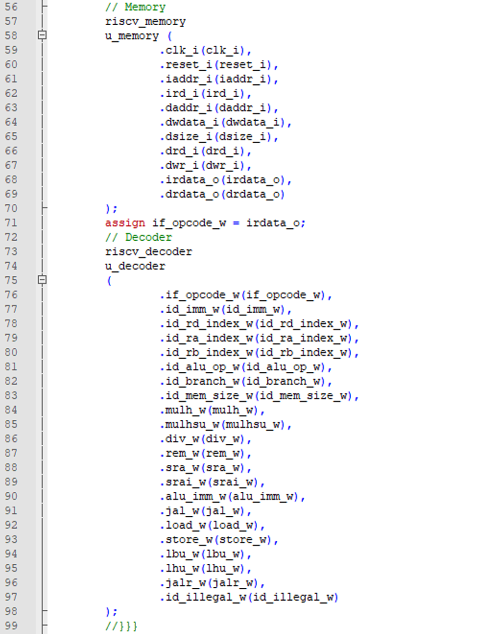
* + Assembly code: add x9, x8, x0
  + Machine code (RISC-V): 00040483
  + Decoder results: rd=9 (x9), ra=8 (rs1/x8), rb=0 (rs2/x0), alu\_op=0 (ALU\_ADD).



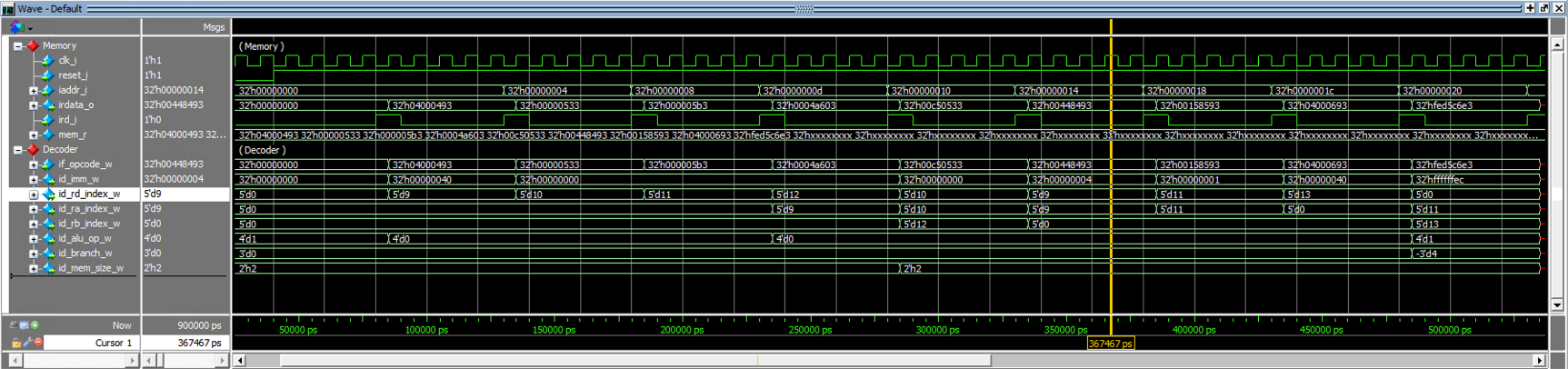
- riscv\_core\_sim\_tb.v



Re-defined if\_opcode\_w reg to wire. ( Initially defined as reg )



- Wave Form



**Problem 4 (2p): (Optional) Bonus**

Briefly explain your code in Problem 1.

1. Memory Initialization

Memory loads firmware and loads data including instructions to the memory.

2. Instruction Memory

Since instruction cannot be written, memory only offers reading instructions from the memory. Because RV32 instruction set consists of 32-bit length instructions, memory just returns the instruction of given address.

3. Data Memory

Since data can be a byte, two bytes(half-word), four bytes(full word) long, memory determines size of data which needs to be written or read by dsize\_r pin. After determining the size of data, if reading data from the memory, first read the full-size word at address and slice the full size word depending on the size of data. If writing data to the memory, first determine which address and slot to write data by given address(daddr\_i) and write the data to the memory.