16-bit pipelined RISC Microprocessor

Write the verilog RTL description for a 16-bit pipelined RISC Microprocessor. The microprocessor is based on the MIPS architecture. Refer to the following for more information if required:

David A. Patterson and John L. Hennessy, “Computer Organization and Design: The hardware/software interface”, Morgan Kaufmann Publishers.

The design should use a five stage pipeline. Each pipeline stage operates over 2 clock cycles. You can clock your pipeline registers directly using the clock or a divide by 2 version of the clock. The interface to the chip is given in the following diagram

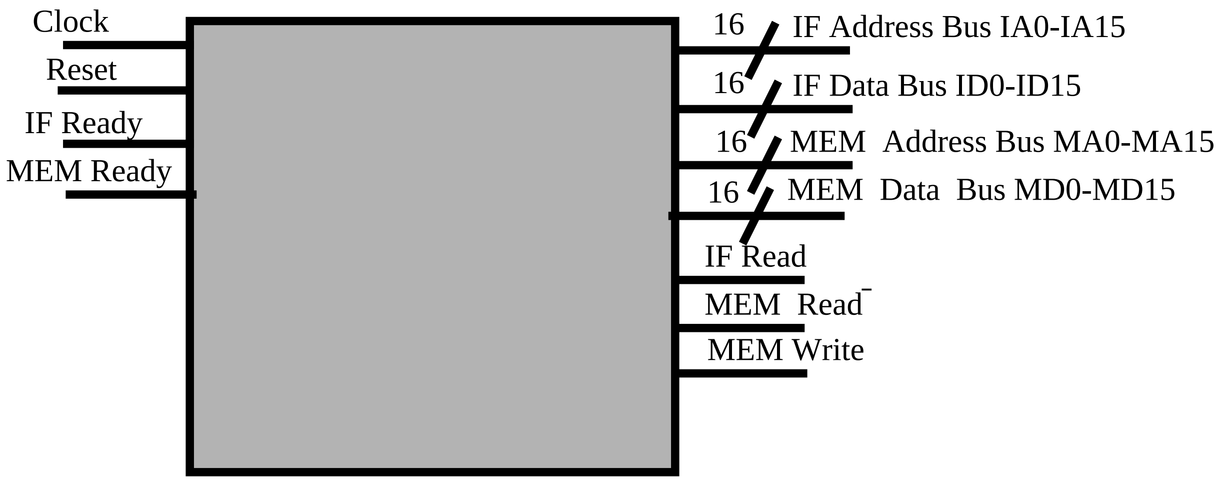
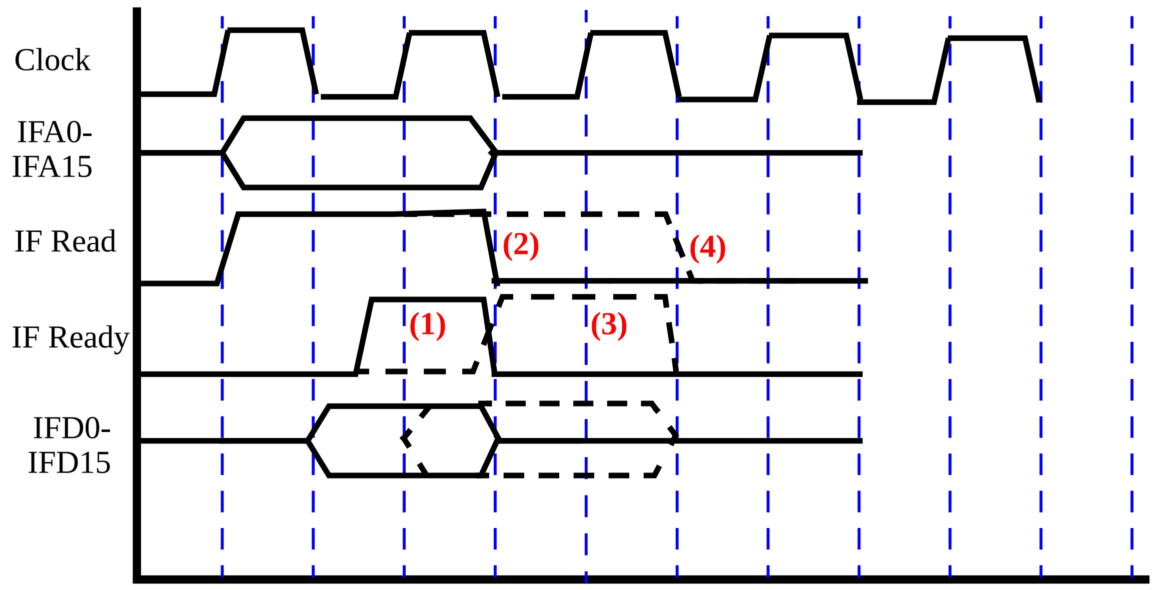


Figure 1

- IF: Instruction Fetch: Send the program counter (PC) address to the instruction memory and read the instruction. Computer the next value of PC=PC +2. Determine the next PC value either using the increment above or using forwarded data from a branch or jump instruction.Timing diagram for the read access is shown below in Figure 2.
- ID: Instruction Decode: The register file is located in this stage. It consists of 8 16-bit registers R0-R7. This stage also contains the sign extension unit for sign-extending immediate fields in the instruction. Data is written from the writeback stage on the first rising edge of clock and read on the second rising edge of the clock. Branch decoding and address calculation are also done here. Data can be forwarded to this stage.
- EX: Execute: This is were all the functional units are located. Data can be forwarded to this stage. Data can be forwarded from this stage
- MEM: Memory Access: Used to access the data memory for load/store instructions. Timing diagram for read is similar to IF read timing. For write, dump address and data on first rising edge. Turn on write on the falling edge, check ready on the next falling edge. If ready is 1, turn off write and remove address and data on next rising edge. Data can be forwarded from this stage
- WB: Write Back: Write the data back into the register file on the first rising edge of clock.



(1) Check ready on second rising edge of clock, if 1 latch the data in internal registers.

(2) Turn off read on the next falling edge, transfer data to pipeline register on next rising edge

(3) If ready was zero, stall by 1 clock cycle and sample ready again at next rising edge

(4) Repeat step 2 with one cycle stall inserted

Figure 2

The process should assume that a branch is always not take, i.e. you can fetch the next instruction after a branch instruction. If you find in the decode stage that the branch is taken, then you will flush the IF stage, i.e. insert a NOP instead of the fetched instruction. Keep in mind the setup and hold restrictions of flip-flops and latches during the design phase. Insert delay elements when there is no logic between two flip-flops. This arises in situations when something is just passed from one pipeline register to the next without any processing e.g. the store data read in ID from registers is passed directly in the EX stage without any processing. There could be possible timing violation between the ID/EX pipeline register and EX/MEM pipeline register. As explained before your processor should be able to forward data as required between various stages to avoid data hazards. The pipeline should be able to stall for hazards that cannot be solved using data forwarding.

Instruction Formats:
5 bit opcode, plus last 2 bits in the opcode for different variants.
3 bit operand fields (Rd, Rs1, Rs2) to specify one of the 8 general purpose registers.
Immediates are restricted in length to the number of bit positions remaining in a 16 bit instruction word.
ALL instructions are 16 bits in length.
Shift immediates can be restricted to 2 bits.

Name	Opcode 5:2 bits	Format	Notes
SEQ	00000:00	SEQ Rd Rs1 Rs2 00	Set if Equal: Set register Rd to 1 if the values in Rs1 and Rs2 are equal, else set to 0.
SGT	00000:01	SGT Rd Rs1 Rs2 01	Set if Greater then: Set register Rd to 1 if Rs1 > Rs2, else set to 0.
SLT	00000:10	SLT Rd Rs1 Rs2 10	Set if Less then: Set register Rd to 1 if Rs1 < Rs2, else set to 0.
SNE	00000:11	SNE Rd Rs1 Rs2 11	Set if Not Equal: Set register Rd to 0 if the values in Rs1 and Rs2 are equal, else set to 1.
ADD	00001:00	ADD Rd Rs1 Rs2 00	Add registers Rs1 and Rs2 and store the result in Rd.
SUB	00001:01	SUB Rd Rs1 Rs2 10	Compute Rs1 - Rs2 and store the result in Rd.
NEG	00001:11	NEG Rd Rs1 000 11	Put 2’s complement of Rs1 into Rd
ADDI	00010	ADDI Rd Rs #	Store in register Rd the value Rs + the 5-bit sign-extended immediate.
SUBI	00011	SUBI Rd Rs #	Store in register Rd the value Rs - the 5-bit sign-extended immediate.
AND	00100:00	AND Rd Rs1 Rs2 00	Store in register Rd the bitwise AND of Rs1 and Rs2.
OR	00100:01	OR Rd Rs1 Rs2 01	Store in register Rd the bitwise OR of Rs1 and Rs2.
XOR	00100:10	XOR Rd Rs1 Rs2 10	Store in register Rd the bitwise XOR of Rs1 and Rs2.
NOT	00100:11	NOT Rd Rs 000 11	Store in register Rd the bitwise complement Rs.
ANDI	00101	ANDI Rd Rs #	Store in register Rd the bitwise AND of Rs and the 5-bit sign-extended immediate.
ORI	00110	ORI Rd Rs #	Store in register Rd the bitwise OR of Rs and the 5-bit zero-extended immediate.
SRA	00111	SRA Rd Rs 000 #	Shift Right Arith: Store in register Rd the sign-extended value of Rs shifted to the right by the 2-bit immediate. (An immediate of 0 does not shift the operand).

Name	Opcode 5:2 bits	Format	Notes
SRL	01000	SRL Rd Rs 000 #	Shift Right Logic: Store in register Rd the zero-extended value of Rs shifted to the right by the 2-bit immediate.
SLL	01001	RLL Rd Rs 000 #	Shift Left Logic: Store in register Rd the zero-extended value of Rs shifted to the left by the 2-bit immediate.
LW	01010	LD Rd Rs #	Load the 16-bit value at memory address Rs + sign extended immediate into Rd. (You can assume that address is an even number).
SW	01011	SW Rd Rs #	Store to memory at address Rd+ sisgn extended immediate the value in Rs. (You can assume that address is an even number).
LBI	01100	LBI Rd #	Load Byte Immediate: Store the 8-bit sign-extended immediate into register Rd after sign extending it.
LBIU	01101	LBIU Rd #	Load Byte Immediate Unsigned: Write the 8-bit zero-extended immediate into register Rd.
LHI	01110	LHI Rd #	Load High Immediate: Write the 8-bit immediate into the upper 8 bits of register Rd and preserve the low order 8 bits.
LLI	01111	LLI Rd #	Load Low Immediate: Write the 8-bit immediate into the lower 8 bits of register Rd and preserve the higher order 8 bits.
BEQZ	10000	BEQZ Rs #	Branch on Equal to Zero: Set PC to PC + 8-bit sign-extended immediate if the value in register Rs is zero.
BNEZ	10001	BNEZ Rs #	Branch if Not Zero: Set PC to PC + 8-bit sign-extended immediate if the value in register Rs is non-zero.
SGE	10010:00	SGE Rd, Rs1, Rs2 00	Set if Greater than Equal: Set register Rd to 1 if Rs1 >= Rs2, else set to 0.
SLE	10010:01	SLE Rd, Rs1, Rs2 00	Set if Less than Equal: Set register Rd to 1 if Rs1 <= Rs2, else set to 0.
J	10011	J #	Jump: Set the PC to PC + 11-bit sign-extended immediate.

Name	Opcode 5:2 bits	Format	Notes
JR	10100	JR 000 Rs	Jump to Register: Set the PC to the value in register Rs.
JAL	10101	JAL #	Jump and link: Set the PC to PC + 11-bit sign-extended immediate. Save next PC i.e. PC+4 value at R0.
JALR	10110	JALR Rd Rs	Jump and Link Register: Save the current value of PC i.e. PC +4 to register Rd and set PC to value in register Rs
MOVE	10111	MOVE Rd Rs	Copy value in Rs into Rd
Unused	11000-11110	Expandable instructions	Required for 641 students. Should include 8-bit multiply and 8-bit divide
NOP	11111	NOP	Do nothing.