COL216: ASSIGNMENT 10

DIPEN KUMAR (2018CS50098)
UMESH PARMAR (2018CS5424)

Simulating a variable delay in Pipelined Processor with forwarding-
In this assignment we have extended the MIPS processor of assignment 9 by introducing HIT rate to access memory in 1 clock cycle else N clock cycle in the pipelined processor.
(Ignoring branch hazards)

Specifications –

1. This is a five stage pipelined processor with ALU and Memory forwarding for data hazard optimization.
2. In case of branch hazard we stall the processor.
3. Two different catches for instruction catch and data catch, each with single read or write port is used for countering structural hazard.

Stages in details –

1. Stage 1 [instruction fetching] – instruction is fetched form Instruction memory and stored in IF/ID pipelined register.
2. Stage 2 [instruction decoding] – instruction in IF/ID pipelined register is decoded for following –
   - add – rd, rs, rt and command for addition is noted and stored in ID/EXE pipelined register.
   - sub – rd, rs, rt and command for subtractions is noted and stored in ID/EXE pipelined register.
   - sll – rd, rt, shamt and command for shift left logical is noted and stored in ID/EXE pipelined register.
   - srl – rd, rt, shamt and command for shift right logical is noted and stored in ID/EXE pipelined register.
   - lw – rt, rs, offset and command for load word is noted and stored in ID/EXE pipelined register.
   - sw – rt, rs, offset and command for store word is noted and stored in ID/EXE pipelined register.
   - jr – rs and command for jump register is noted and stored in ID/EXE pipelined register.
   - j – target and command for jump is noted and stored in ID/EXE pipelined register.
   - jal – target and command for jump and link is noted and stored in ID/EXE pipelined register.
   - beq – rs, rt, offset and command for branch on equal is noted and stored in ID/EXE pipelined register.
   - bne – rs, rt, offset and command for branch on not equal is noted and stored in ID/EXE pipelined register.

- ➢ blez – rs, offset and command for branch on less than equal zero is noted and stored in ID/EXE pipelined register.
- ➢ bgtz – rs, offset and command for branch on greater than equal zero is noted and stored in ID/EXE pipelined register.

3. Stage 3 [execution] – instruction decoded in stage 2 in previous cycle is fetched from ID/EXE pipelined register and is executed using ALU for following commands –
   - ➢ add – rs and rt is added and result with command and rd is stored in EXE/MEM pipelined register.
   - ➢ sub – rt is subtracted from rs and result with command and rd is stored in EXE/MEM pipelined register.
   - ➢ sll – rt is shifted left logically by shamt and result with command and rd is stored in EXE/MEM pipelined register.
   - ➢ srl – rt is shifted right logically by shamt and result with command and rd is stored in EXE/MEM pipelined register.
   - ➢ lw – rs is added with offset to get address and result with command and rt is stored in EXE/MEM pipelined register.
   - ➢ sw – rs and offset is added to get address and result with command and rt is stored in EXE/MEM pipelined register.
   - ➢ jr – set next instruction address to rs.
   - ➢ j – set next instruction address to target.
   - ➢ jal – set next instruction address to target and command and next instruction address is stored in EXE/MEM pipelined register.
   - ➢ beq – set next instruction address to offset + next instruction address if rs is equal to rt
   - ➢ bne – set next instruction address to offset + next instruction address if rs is not equal to rt
   - ➢ blez – set next instruction address to offset + next instruction address if rs is less than or equal to 0.
   - ➢ bgtz – set next instruction address to offset + next instruction address if rs is greater than or equal to 0.
4. Stage 4 [memory] – data memory is fetched according to instructions stored in EXE/MEM pipelined register.
   - ➢ add – result, command and rd is stored in MEM/WB pipelined register.
   - ➢ sub – result, command and rd is stored in MEM/WB pipelined register.
   - ➢ sll – result, command and rd is stored in MEM/WB pipelined register.
   - ➢ srl – result, command and rd is stored in MEM/WB pipelined register.
   - ➢ lw – data at address is fetched in data catch and it is stored in MEM/WB piplined register along with command and rt. (it may require multiple one or N clock cycle depending on HIT or MISS respectively)
   - ➢ sw – register data at rt is stored in data memory at address. (it may require multiple one or N clock cycle depending on HIT or MISS respectively)
   - ➢ jal – command and next instruction address is stored in MEM/WB pipelined register.
5. Stage 5 [write back] – register is written back according to instructions stored in MEM/WB pipelined register.
   - ➢ add – result is written in register at rd.
   - ➢ sub – result is written in register at rd.

- sll – result is written in register at rd.
- srl – result is written in register at rd.
- lw – data from data memory present in MEM/WB pipelined register is written in register at rt.
- Jal – next instruction address is written in register 31.

INPUT –

1. Instruction catch is stored in text.txt
2. Data catch is stored in data.txt

OUTPUT –

1. Register file state after every clock cycle in register.txt
2. Data catch state after every clock cycle in memory.txt
3. Instruction that is active in each stage of the pipeline after every clock cycle on console.
4. Instruction counts, clock cycle counts and IPC for the program at the end on console.

IN SHORT –

When a miss occurs while processing lw and sw instructions which we get to know in MEM stage- I am stalling N-1 cycle. Previous instruction which is in WB stage will continue execution while next following instructions will wait for N-1 clock cycles in their respective stages. Meanwhile “lw” or “sw” instruction which is in MEM stage is fetching data from data catch, that means it will also remain N-1 cycle more in MEM stage.
When it is hit, it will continue processing as Assignment 9 processor.