CDA 4205 Computer Architecture

Assignment 6: Single-Cycle Processor Implementation

1. (15 pts) Describe the effect that a single stuck-at-0 fault (i.e., the signal is always 0 regardless of what it should be) would have for the signals shown below, in the single-cycle Datapath. Which instructions, if any, will not work correctly? Explain why.

Consider each of the following faults separately:

1. RegWrite = 0

**This means that no register can be written in the register file. All R-type instructions with destination register Rd, such as ADD and SUB, and all I-type instructions with destination register Rt, such as ORI and LW, will not work because these instructions will not be able to write their results to the register file.**

1. RegDst = 0

**This means that RegDst is stuck at Rt and can never be Rd. All R-type instructions with destination register Rd, such as ADD and SUB, will not work because these instructions will not be able to write their results to Rd. Instead, they will write their results to the second source register Rt.**

1. ALUScr = 0

**This means that ALU is stuck to have its second operand coming from the register file, and can never be the immediate constant encoded inside the instruction. All I-type instructions with a 16-bit immediate constant, such as ORI, BEQ, LW, and SW, will not work because these instructions will not be able to use the 16-bit immediate constant encoded inside the instruction.**

1. MemtoReg = 0

**This means that the value written back to the register file is always the ALU result, and can never be the value read from data memory. The load instructions, such as LW, will not work because these instructions will not be able to write back the value read from data memory into the register file.**

1. Branch = 0

**This means that the Branch control signal will never indicate the presence of a branch instruction. The branch instructions, such as BEQ, will not work because these instructions will not be able to branch (branch is never taken) even when the branch condition is true.**

1. (15 pts) Repeat question 1 but this time consider stuck-at-1 faults (the signal is always 1).

**Solution:**

1. RegWrite = 1

**This means that all instructions will write ‘some value’ in the register file. Store instructions, such as SW, branch instructions, such as BEQ, and jump instructions will not work properly. These instructions are not supposed to modify the register file. However, when RegWrite = 1 then Store, Branch, and Jump instructions will modify the register file.**

1. RegDst = 1

**This means that RegDst is stuck at Rd and can never be Rt. All I-type instructions with destination register Rt, such as ORI and LW, will not work because these instructions will not be able to write their results to Rt. Instead, they will write their result to some arbitrary register Rd whose number is encoded in the upper 5 bits of immediate constant.**

1. ALUScr = 1

**This means that the ALU is stuck to have its second operand coming from the immediate constant encoded inside the instruction. All R-type instructions with a second source register Rt, such as ADD and SUB, will not work because these instructions will not be able to use the second operand read from the register file. Instead the datapath will assume that the R-type instruction was an I-type instruction, and will assume the second ALU operand to be an immediate constant.**

1. MemtoReg = 1

**This means that the value written back to the register file is always the value read from data memory, and can never be the ALU result. All instructions that produce an ALU result, such as ADD, SUB, and ORI, will not work because these instructions will not be able to write back the ALU result into the register file.**

1. Branch = 1

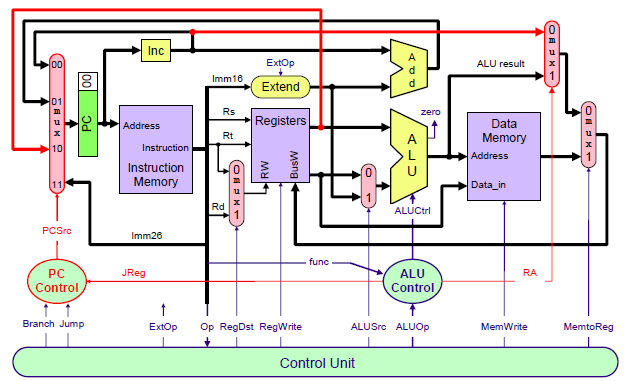
**This means that the Branch control signal will always indicate the presence of a branch instruction. All non-branch instructions, such as ADD, SUB, ORI, LW, and SW, will not work because these instructions will be able to branch. However, these instructions were never supposed to branch.**

1. (15 pts) We wish to add the instruction **jalr** (jump and link register) to the single-cycle datapath. Add any necessary datapath and control signals and draw the result datapath. Show the values of the control signals to control the execution of the **jalr** instruction.

The jump and link register instruction is described below:

**jalr rd, rs # rd = pc + 4 , pc = rs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **op6 = 0** | **rs5** | **0** | **rd5** | **0** | **Funct6 = 0x9** |



**The necessary changes to the datapath and control:**

**For the datapath, we need a bigger 4-input multiplexer at the input of the PC. The first input is used to increment the PC. The second input is used for taken branches, where the branch target is PC-relative. The third input is used to jump register, where the input to the PC comes from a general-purpose register, and the fourth input is used for jump instructions.**

**For the implementation of the JALR instruction: to jump to register ‘Rs’, we need to add a path from the output of register Rs (first ALU input) back to the PC multiplexer input. PC control unit needs to be updated by adding an input control signal JReg (Jump Register) to select PC according to the value of register Rs. JReg is generated by the ALU control unit, since JALR is a R-type instruction and JReg depends on the function field only. When JReg is equal to ‘1’, PCSrc (PC control unit output control signal) will be '10' to select the value of register Rs as input to PC.**

**Also, we need to store PC+4 in register Rd. To accomplish this, we need another multiplexer to select between the incremented PC, the ALU result and data memory out, to be placed on BusW. Also, we need to add a path from the output of the incremented PC to the input of this new multiplexer. A control signal ‘RA’ (Return Address) is needed to select between the incremented PC and the ALU result. The MemtoReg multiplexer selects between the output of the ‘RA’ multiplexer and the Data Memory output to place on BusW.**

**The main control signals for the JALR instruction are the same for other R-type instructions, such as ADD and SUB. The ALU Control signals for the JALR instruction require JReg = 1, RA = 1 and ALUCtrl is a don't care. These control signals are shown in the table below:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RegDst** | **RegWrite** | **ExtOp** | **ALUSrc** | **MemRead** | **MemWrite** | **MemtoReg** | **ALUCtrl** | **J** | **Beq** | **Bne** | **RA** | **JReg** |
| **Rd = 1** | **1** | **X** | **X** | **0** | **0** | **0** | **XXXX** | **0** | **0** | **0** | **1** | **1** |

1. (15 pts) Suppose we add the multiply and divide instructions. The operation times are as follows:

Instruction memory access time = 190 ps, Data memory access time = 190 ps,

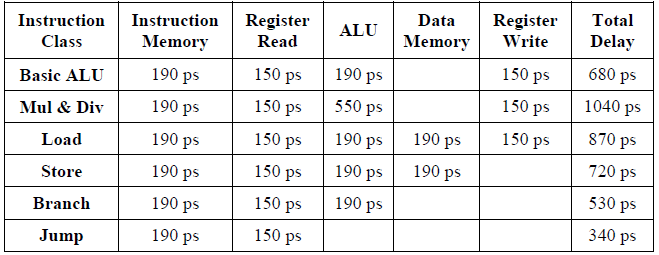
Register file read access time = 150 ps, Register file write access = 150 ps

ALU delay for basic instructions = 190 ps, ALU delay for multiply or divide = 550 ps

Ignore the other delays in the multiplexers, control unit, sign-extension, etc.

Assume the following instruction mix: 30% ALU, 15% multiply & divide, 20% load, 10% store, 15% branch, and 10% jump.

1. What is the total delay for each instruction class and the clock cycle for the single-cycle CPU design?



**Clock cycle = max delay = 1040 ps.**

1. Assume we fix the clock cycle to 200 ps for a multi-cycle CPU, what is the CPI for each instruction class and the speedup over a fixed-length clock cycle?

**Solution:**

**CPI for Basic ALU = 4 cycles**

**CPI for Multiply & Divide = 6 cycles**

**CPI for Load = 5 cycles**

**CPI for Store = 4 cycles**

**CPI for Branch = 3 cycles**

**CPI for Jump = 2 cycles**

**Average CPI = 0.3 \* 4 + 0.15 \* 6 + 0.2\* 5 + 0.1 \* 4 + 0.15 \* 3 + 0.1 \* 2 = 4.15**

**Speedup of multi-cycle over single-cycle = (1040 \* 1) / (200 \* 4.15) = 1.253**

* **Submission Requirements**
* Your solutions must be in a single file with a file name yourname-hw6.
* If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
* Only submissions via the link on Canvas where this description is downloaded are graded. Submissions to any other locations on Canvas will be ignored.
* Late submissions are accepted for a maximum of 3 late days with 20% assignment credit off as late penalization. Assignments submitted after 3 late days will not be accepted.