CE1921 LABORATORY PROJECT

SUMMARY

Instruction set architecture describes the programmer's view of the machine. This level of computer blueprinting allows design engineers to discover expected features of the circuit including:

- data processing instructions like ADD, SUB, AND, EOR, etc.,
- memory load-store instructions like LDR, STR, etc.,
- branch instructions with both conditional and unconditional flow control,
- the names and sizes of the registers provided for computational storage and flow control,
- the size of data memory provided for longer term storage during program execution, and
- the binary encodings for each instruction.

These features are then used by design engineers as they choose components to organize together into a working circuit. Multiple micro-architectures are possible for any given instruction set architecture because different design engineers can choose different components and different organizational strategies when implementing the features. In the end, however, any micro-architecture design must implement the features described by the instruction set architecture. One organizational strategy decision that leads to different micro-architectures is the number of clock periods used per instruction. The three common clock-period strategies are called ***single-cycle***, ***multi-cycle***, and ***pipelined***.

- Single-cycle processors use one clock-period per instruction and the clock-period is set by the total circuit delay required to execute the slowest instruction. This is a disadvantage as faster instructions cannot execute more quickly. The advantage, however, is straightforward control circuitry.
- Multi-cycle processors use multiple clock-periods per instruction and each instruction uses the minimum number of clock period required for its execution. This allows faster instructions like ADD that do not access data memory to avoid the unnecessary delay of the data memory stage. Thus, the advantage is speed for faster instructions. The disadvantage is more complex control because a finite state machine controller must be built to coordinate control signals across multiple clock periods.
- Pipelined processors exploit instruction level parallelism to allow multiple instructions to be in execution at the same time. This is accomplished by adding state registers between the instruction fetch, instruction decode, execute, memory access, and write-back stages of the circuit. Performance improves because multiple instructions are executing in the processor circuit during each clock period. The challenge is keeping the pipeline full of instructions that can be executing simultaneously.

This multi-week project requires students to design and simulate an ***pipelined processor***. Over the next two weeks, the basic single cycle processor must be transformed into a pipelined processor that implements forwarding for data hazard management and early-branch detection for control hazard management. Branch prediction is not implemented.

CE1921 LABORATORY PROJECT

EXERCISES

1. Transform the single cycle processor to a basic pipeline that does not manage control or data hazards. Thus, this pipeline cannot stall or flush. It is capable of correctly executing only long streams of ARMv4 instructions that do not have data hazards or any branches.
2. Extend your basic pipeline to a pipeline with data hazard management implemented with forwarding signals at the ALUSRCA and ALUSRCB multiplexers. Forwarding control can be implemented within the ALUSRCA and ALUSRCB signal equations of the decode stage controller or a separate forwarding controller can be constructed. This pipeline does not flush. Thus, it is capable of handling long streams of ARMv4 instructions that have data hazards but no branches.
3. Extend your DHM-pipeline to an advanced pipeline that manages both data hazards and branch hazards. This complete pipeline (CPIPE) handles all data hazards, including those caused by LDR, as well as branch control hazards caused by B, BEQ, BNE, and BL to leaf subroutines (subroutines that do not call other subroutines).

These exercises must be completed over weeks 8, 9, and 10. Students can elect to work toward any of the goals with lab points distributed in the following way.

PIPELINE	GRADE LEVEL	MAXIMUM LAB POINTS POSSIBLE
NONE	no work	0
BASIC	C-level work	70
DHM PIPE	B-level work	85
CPIPE	A-level work	100

DELIVERABLES AND DUE DATE

1. This laboratory must be demonstrated to your instructor no later than Friday of week 10.
2. Screenshots of VHDL simulation verifying results must be included in your submission packet. Add written comments that demonstrate how you know your simulation is correct.
3. Additional materials may be required by your instructor in your submission packet.
4. **Due date:** This laboratory must be demonstrated to the instructor no later than the end of week 10.