1 Project 3: Non-Pipelined Control Unit

Due: 3/6/2018
Points: 50

1.1 The Objective

The principle objective here is to demonstrate that you understand the basic operation of the control unit within a simple processor. You can continue to use a CAD tool for the gate level circuit design or you can draw the circuits by hand. CAD tool use and simulation are strongly recommended, but not required. At your discretion, you can work on this project in teams of two individuals. It is solely your responsibility to form a team. I am not responsible for any team members performance issues, including a member's withdrawal from the class.

1.2 The Task

Design a single bus gate level implementation of a machine that implements the following instruction set. You are to use a hardwired control unit. The project is worth a total of 50 total points. 40 of these points will be based upon the correctness of the design and the remaining 10 points will be determined by (*documented*) optimizations that you make to the design. In order to achieve credit, you must document your optimizations, describe the trade offs of each optimization, and justify your selected optimizations. Optimizations achieved by pipelining will not count, that is the objective of Project 4.

In addition to solving this problem, you are expected to deliver documentation to your solution that is well-organized, modular, and thoroughly described. You cannot simply turn in a circuit design and expect credit. Part of your challenge is to discover a method to deliver a documented solution that is easy to study, digest, and understand. You will lose points if you do not develop a suitable presentation for your design solution.

2 Grading Specifics

Project 3 will be graded as follows.

- Correctness:
  - (10 points) overall description/documentation of solution. basically an english description of the project solution (not counting the optimizations discussion).
  - (10 points) the correctness of the control signals for each state in the system
  - (10 points) the design and organization of the state machine and it's implementation and the hardware elements (registers, alu, etc)
  - (10 points) the use of the state machine components to generate control signals
- Optimization:
  - 10 points for documented optimizations

3 Processor Specification

3.1 Basic Characteristics

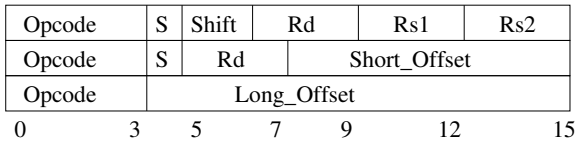
- a word size of 16-bits
- memory address/data bus size of 16-bits
- byte addressable memory
- 64K byte main memory
- a 16-bit program status word (PSW) with status bits. The first two bits are the condition code bits Z and N; these bits are conditionally controlled and denote the results of comparisons of the instruction result to

the values zero (Z=1-equality to zero; N=1-less than zero). In addition, a third bit denotes execution in either of privileged or user mode (some operations are prohibited in user mode). The remaining bits control operations/constraints in the memory space that are not addressed in this project.

- 16 instructions, 2 of which can be executed only in privileged mode. Attempt to execute these 2 instructions in user mode will cause a program check violation.
- 8 16-bit General Purpose Registers (Reg). Register 0 (Reg[0]) always holds the value 0 no matter what value is assigned to it.
- a 16-bit program counter (PC) which is also Reg[7]
- a 16-bit count-down timer that causes a timer interrupt when it reaches zero provided the machine is executing in user mode. Timer interrupts are ignored when executing in privileged mode.
- 2's complement number representation

3.2 Instruction Format

All instructions are conditionally executed. This is described more fully below. In this instruction set, there are three instruction formats organized as follows:



`Rd/Rs1/Rs2` specify one of the general purpose registers, `S` controls the setting of the condition codes (described below), and the use of the offset fields is defined in the instruction set definition. The `Shift` bits in the instruction cause the register value at `Rs2` to be left shifted by the corresponding value (0, 1, 2, or 3).

3.3 Instructions

The instruction set consists of 16 instructions shown in Table 1. The notation `GPR[]/MM[]` denotes respectively the register contents/memory contents.

The condition codes are conditionally set by the result from the first and second instruction formats. The instruction bit `IR.S` determines if the execution of the instruction should set the condition code (1 – Yes, 0 – No). If set, the condition code bit (N and Z) should be set based on the value resulting from the operation.

When in user mode (i.e., the PSW privileged bit (P) is not set), only the first 14 instructions can be executed. All 16 instructions can be executed in privileged mode. Attempting to execute a privileged instruction when in user mode signals a program check violation.

3.4 Exceptions

3.4.1 Program Check Violations

When in user mode (*i.e.,* the PSW privileged bit (P) is not set), only the first 14 instructions can be executed. All 16 instructions can be executed in privileged mode. Attempting to execute a privileged opcode when in user mode signals a program check violation. A program check violation causes the machine to swap the PC, and PSW as follows:

1. MM[0] = PSW
2. MM[2] = PC
3. PSW = MM[4]
4. PC = MM[6]

3.4.2 Timeout

There exists a count-down timer in the system that interrupts execution of instructions when executing in user mode. When this counter reaches zero, it triggers an internal state bit.

Name	Opcode	Description
ADD	0	GPR[Rd] = GPR[Rs1] + left_shifted(GPR[Rs2],IR.Shift)
SUB	1	GPR[Rd] = GPR[Rs1] - left_shifted(GPR[Rs2],IR.Shift)
AND	2	GPR[Rd] = GPR[Rs1] **and** left_shifted(GPR[Rs2],IR.Shift)
SHL	3	GPR[Rd] = shift_left(GPR[Rs1]) by left_shifted(GPR[Rs2],IR.Shift)$_{3-0}$
SHRA	4	GPR[Rd] = shift_right(GPR[Rs1]) by left_shifted(GPR[Rs2],IR.Shift)$_{3-0}$
OR	5	GPR[Rd] = GPR[Rs1] **or** left_shifted(GPR[Rs2],IR.Shift)
NOT	6	GPR[Rd] = **not** MM[PC + Short_Offset]
LD	7	GPR[Rd] = MM[PC + Short_Offset]
ST	8	MM[PC + Short_Offset] = GPR[Rd]
BRN	9	if CC.N then PC = PC + Long_Offset
BRZ	10	if CC.Z then PC = PC + Long_Offset
BR	11	PC = PC + Long_Offset
JSR	12	GPR[Rd] = PC; PC = PC + Short_Offset
RTS	13	PC = GPR[Rd] + Short_Offset
CLK	14	Set timer to MM[PC + Long_Offset]
LPSW	15	PSW = MM[PC + Long_Offset]

All uses of Short_Offset and Long_Offset are sign extended; left_shifted(GPR[Rs2],IR.Shift)$_{3-0}$ denotes the low order 4 bits of the operand value; the left shift is logical; and the right shift is arithmetic.

Table 1: Instruction and their Semantics

This internal state bit is reset when a new value is loaded into the clock (by the CLK instruction). A timeout exception interrupt occurs when the internal state bit is set and the control unit is at an instruction boundary (between instructions). The effects of the interrupt are to modify the main memory, PC, and PSW in the following way:

1. MM[8] = PSW
2. MM[10] = PC
3. PSW = MM[12]
4. PC = MM[14]

When executing in privileged mode, the countdown timer has no effect.

4 Restrictions

1. You may use any number of internal registers to hold intermediate values. You *must* restrict yourself to the single bus paradigm — *no* point-to-point connections are allowed.
2. You may use a constant ROM in this design provided it contains 8 or fewer constants. You do not have to develop a gate level description of the ROM, but you must provide an informal description of its operation. You may assume that it operates sufficiently fast to provide data to the bus in the same clock cycle as it is needed.
3. Assume that all memory operations are asynchronous. You *do not* have to develop a

gate level implementation of the main memory; however, you do have to give an informal description of its interface.

4. You must develop gate level descriptions of all components except: multiplexers, demultiplexers, encoders, decoders, and flip-flops. You can use `and`, `or`, `not`, `xor`, `nand`, and `nor` logic gates in your solution.
5. Ignore the details regarding the implementation of the count-down timer and define only its interface. You must however, use the system clock to pulse the count-down timer.

5 The Report

Your report must clearly document your design solution. The documentation should include informal descriptions of the various components in your solution. It should should include the control signals written out in a logical structure that can be meaningfully understood by someone not part of the project team. The synthesis of the control unit circuit should be presented, including the finite state automata and the next state table.

There are no limits on the length of the report. Good luck.

6 Change Log

Revision 0: 2/6/2018

- Initial Version