

University of California, Berkeley
 College of Engineering
 Computer Science Division
 Electrical Engineering and Computer Science Department

Computer Science 150
 Spring 2001

R. H. Katz

Problem Set #10/11 (Assigned 12 April, Due 27 April)

1. Introduction

Your task is to complete the high level design of a very simple 4-bit computer. This will involve designing the datapath and a control state machine to the state diagram level to implement the instruction set. This is a very good example of the kind of comprehensive question you can expect to find on the final examination!

The machine has a single accumulator (R_0), four general purpose registers (R_0, R_1, R_2, R_3), two logical shift instructions (LSL--logical shift left, LSR--logical shift right), an arithmetic shift instruction (ASR--arithmetic shift right with sign extend), a rotating shift instruction (RSL--rotating shift to the left), three register-register instructions (ADD, NAND, XFER), two register-memory instructions (LOAD, STOR), and a conditional branch instruction (BRZ). These instructions are encoded in from one to three 4-bit instruction parcels. The machine can address 256×4 -bit words.

2. Instruction Set

Instructions are encoded in from one to three 4-bit words. Arithmetic instructions are encoded in a single word: two opcode bits (X_3X_2) and two operand bits (X_1X_0). All arithmetic instructions use R_0 as an accumulator. The operand bits specify one of the four on-chip registers. The LSR, LSL, ASR, and RSL instructions involve a shifter subsystem, and use the operand bits to encode the four different operations.

Arithmetic Instructions

<i>Op Code (Binary)</i>	<i>Op Code (Symbolic)</i>	<i>Function</i>
00 X_1X_0	ADD	$R[0] \leftarrow R[0] + R[X_1X_0]$
01 X_1X_0	NAND	$R[0] \leftarrow R[0] \text{ NAND } R[X_1X_0]$
10 00	LSR	$R[0]<3>, R[0]<2:0> \leftarrow 0, R[0]<3:1>$
10 01	LSL	$R[0]<3:1>, R[0]<0> \leftarrow R[0]<2:0>, 0$
10 10	ASR	$R[0]<3>, R[0]<2:0> \leftarrow R[0]<3>, R[0]<3:1>$
10 11	RSL	$R[0]<3:1>, R[0]<0> \leftarrow R[0]<2:0>, R[0]<3>$

An opcode of 11 represents an extended instruction, and the remaining two bits of the word specify one of the four remaining instructions. An instruction word of 1100 encodes a register transfer instruction. The operands are encoded in the following word: the high order two bits identify the destination register, while the low order two bits identify the source.

Register Transfer Instruction

<i>Op Code (Binary)</i>	<i>Op Code (Symbolic)</i>	<i>Function</i>
1100 $Y_3Y_2Y_1Y_0$	XFER	$R[Y_3Y_2] \leftarrow R[Y_1Y_0]$

The remaining three instructions are encoded in three 4-bit words: 1 word for the extended op code, and two words to identify the target address (memory extends from location 0 through location 255_{10}).

Memory Reference Instructions

<i>Op Code (Binary)</i>	<i>Op Code (Symbolic)</i>	<i>Function</i>
1101 $Y_3Y_2Y_1Y_0 Z_3Z_2Z_1Z_0$	LOAD	$R[0] \leftarrow \text{MEM}[Y_3Y_2Y_1Y_0 Z_3Z_2Z_1Z_0]$
1110 $Y_3Y_2Y_1Y_0 Z_3Z_2Z_1Z_0$	STOR	$\text{MEM}[Y_3Y_2Y_1Y_0Z_3Z_2Z_1Z_0] \leftarrow R[0]$
1111 $Y_3Y_2Y_1Y_0 Z_3Z_2Z_1Z_0$	BRZ	IF $R[0]=0$ THEN $\text{PC} \leftarrow Y_3Y_2Y_1Y_0 Z_3Z_2Z_1Z_0$

3. Memory Interface

The processor has the following input/output connections to the outside world: RESET (on reset, the PC is set to ZERO), clock, bidirectional data lines $D_3D_2D_1D_0$, and address lines $A_7A_6A_5A_4A_3A_2A_1A_0$. The interface to memory is as described in your textbook: the processor generates a $\overline{\text{CS}}$ (chip select) and $\overline{\text{WE}}$ (write enable) signal to control the RAM directly. We make the significant simplifying assumption that the processor clock period will exceed the RAM access time (i.e., the memory is a 0-Wait State Memory). See Figure 1.

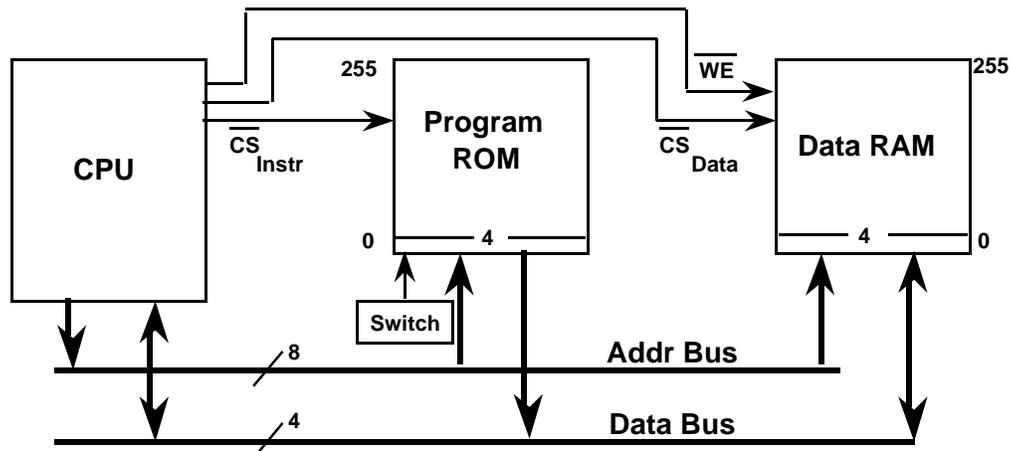


Figure 1: CPU/Memory Interface

The processor should be implemented with a program ROM and a data RAM. Instructions are always fetched from the ROM.

To load/examine program data, the processor should also include a simple “operator’s console” of status LEDs and input switches. It will include a GO push button and a READ/ $\overline{\text{WRITE}}$ switch to load or examine the contents of RAM, along with a switch input to place the processor in a RUN/ $\overline{\text{EXAMINE}}$ mode. When in RUN mode, the processor executes the normal instruction fetch/decode/execute cycles. The operator’s console is inactive, and its memory interface signals are left unasserted. When placed in the $\overline{\text{EXAMINE}}$ mode, the processor completes the execution of the current instruction and enters an idle state, in which the address and data lines are tri-stated, and the memory interface signals are unasserted (driven high). The operator’s console is now active. Eight external switch inputs determine the address of a memory location to be read and output to LEDs or written from four data switches. The READ/ $\overline{\text{WRITE}}$ switch determines the direction of access. The GO button commences a RAM read or write cycle, including appropriate control for the latches and buffers of the console’s datapath.

5. What You Are Expected To Do

1. Design a detailed datapath for this machine, down to the level of registers, functional units, busses, multiplexers, and tri-state gates. Include the operator’s console!

2. Create a list of all register transfer operations supported by your datapath and how these are implemented by detailed control signals.
3. Using your datapath for Part (1) and control signals for Part (2), write down a Moore or Synchronous Mealy Machine state diagram that implements the full instruction set, memory interface, and operator's console. The outputs from the State Machine can be the high-level register transfer operations rather than the detailed controlled signals.