CO1104 (Computer Architecture) Coursework Preparation 3

Module Convenor: Roy Crole
Academic Year: 2023/24
Notes: Exercises on Processor Hardware and Software. Try these out in independent study time, before the CWP tutorial.
The questions refer to notation and ideas from slides 99 to 149
Since these are preparation questions and you will not receive a mark, there is no markscheme.
A few challenging questions are labelled **. You can certainly pass Computer Architecture without doing these questions.
ILOs covered are: Explain, and solve simple problems on, fundamental hardware circuits such as ALUs, multiplexors, and register files.
The topic of "addressing" in Section 2 will not be covered until SU16.

1 Processor Hardware

1. An ALU can perform the operations $+$, $-$, $*$, DIV, &, $|$. It has a control bus of width k, used to select the operations. What is the minimum value of k? Hence write down a table (with six rows) like the ones in the lecture slides, which for each operation specifies a (binary) bus value for its selection. *Hint: How many k-digit binary numbers are there?*

   (* If k were large, how could you calculate the bus width? *Hint: What did you learn in CO1104.*)

2. (a) Draw a picture of a multiplexor when there is just one control line C and two data input *lines* D_0 and D_1. Write down one example of dataflow. Then write down a table of values with columns of C, D_0, D_1 (in) and M (out) and where there are 8 rows to write down (not including the header).

C	D_0	D_1	M
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   (b) Repeat the picture, and give one example dataflow, for a control bus of width two (say C_1C_0) and four data input *busses* D_i where $0 \leq i \leq 3$. This time give a truth table with columns of only C_1, C_0 (in) and M out, stating the value of output M in terms of D_i. The table should have four rows (not including the header).

3. Look at the "&& and || ALU" multiplexor which is on slide 105. Write down the 8-row truth table with input columns labelled C, A, B (in that order) and output R.

4. Look at the multiplexor circuit, slide 108, made from 4 AND gates and an OR gate. Suppose the data input lines satisfy $D_3D_2D_1D_0 = 1001$. Trace the data flow around the circuit first for $C_1C_0 = 00$ and then for 11. Does it behave as you expect, that is, as a multiplexor? (Note: $A \,\&\&\, B \,\&\&\, C = 1 \iff A = 1$ and $B = 1$ and $C = 1$. 4-input $||$ works similarly.)

5. First, look over the circuit for a single bit, slide 113, and make sure you understand how it works by studying the table of values in the slides, and making up some examples with your own dataflow. Now let's think about a 2-bit cell/register, so look at slide 116.

   (a) Draw a picture of a 2-bit cell/register excluding data.

   (b) Using highlighters or coloured ink eg 1, or encircling shapes eg $\boxed{1}$, show the following state changes on the same picture, working out values for ? . What is the name of the initial state?

Din_1	Din_0	RE	WE	Bit_1	Bit_0	$Dout_1$	$Dout_0$	
0	0	0	0	0	0	–	–	eg magenta/no-shape
1	1	0	1	?	?	–	–	eg yellow/square
1	1	1	0	?	?	?	?	eg blue/circle
1	1	0	0	?	?	?	?	eg black/diamond

   Cross out old values and highlight/colour/shape the new values. Make sure the order in which you make your changes is stated clearly if you use different colours/shapes (but please try to use the stated ones).

(c) In a new picture, show the following initial state and state change

Din_1	Din_0	RE	WE	Bit_1	Bit_0	$Dout_1$	$Dout_0$	
1	0	0	1	?	?	?	?	eg magenta/no-shape
1	0	1	0	?	?	?	?	yellow/square

(d) In a new picture, show the following initial state and state changes

Din_1	Din_0	RE	WE	Bit_1	Bit_0	$Dout_1$	$Dout_0$	
0	1	1	0	?	?	1	1	eg magenta/no-shape
0	1	0	0	?	?	?	?	yellow/square
0	0	0	1	?	?	?	?	blue/circle

6. Draw this picture: A register file with 6 empty registers, a `WR` bus, and a `WD` bus. Suppose that 77 is written to register 0: illustrate this on your picture. Do the same for 88 written to register 5. And finally 99 written to 0. (Don't worry about enabling signals.)

7. Now add a single `RR` and a single `RD` bus to your picture. Suppose that 99 and then 88 should be read. Illustrate the dataflow. (Don't worry about enabling signals.)

8. Look at the Register File (RF) on slide 122 that has an ALU to the right. Suppose the RF has ten registers with addresses $0, \dots, 9$. Draw a picture of the registers in a vertical column of rectangles, with addresses, draw a containing box, and draw on the Read Register (x 2), Write Register, Read Data (x 2) and Write Data busses. Suppose the registers contain 100 to 109 respectively.

   (a) If `RR1` $= 6$ and `RR2` $= 8$ draw these denary data on your picture, and the corresponding values of `RD1` and `RD2` assuming reading is on.

   (b) If now `RD1` $= 105$ and `RD2` $= 101$ then write down new values for `RR1` and `RR2` crossing the old ones out.

   (c) Next we execute the ALU using $+$ (using the expected data inputs, discussed in lectures) and write the result into register 9. Give the denary values of `WR` and `WD`.

9. ** You are given two multiplexors, each with two single bit data inputs, and one single bit control; two AND gates; and two OR gates. Construct a circuit which has two (data) input busses $\vec{X} = X_1X_0 \in \mathbb{B}^2$ and $\vec{Y} = Y_1Y_0 \in \mathbb{B}^2$ and a control input $C \in \mathbb{B}$, where the output $R_1R_0 \in \mathbb{B}^2$ can be chosen using control C to be *either* $\vec{X}$ & $\vec{Y}$ *or* $\vec{X} \mid \vec{Y}$, that is

   - $R_1R_0 = (X_1 \;\&\&\; Y_1)\,(X_0 \;\&\&\; Y_0)$
   - $R_1R_0 = (X_1 \;||\; Y_1)\,(X_0 \;||\; Y_0)$

   In the picture of your circuit there should be four data input lines, one control input line, and two output lines. Label it suitably. Write down a control input value of 0 and the corresponding value of R_1 and R_0 on your picture.

10. ** Look over the slide on Addressing that uses a decoder and see if you can understand how it works.

2 Processor Software

1. Draw a picture of the MIPS Register File (similar to the ARM Register File in the slides).

2. (a) MIPS and ARM are *load-store* architectures. Explain briefly what this means.

   (b) Suppose we want to add up five numbers (say 3, 4, 5, 6, 7, each one stored in a main memory cell) and end up with the answer in a main memory cell. Draw some informal pictures of CPU registers and MM cells to explain at an abstract level how this should be done, showing state updates.

   (In the CPU you need registers `$currentNumber` and `$currentSum`, assumed initially to store 0. In MM you need cells for the given data, and for the final sum. Indicate loads and stores with arrows labelled by `lw` and `sw`. You may assume you can use an `add` R, S, T instruction similar to those you have seen before, but there's no need to write any code—only the state updates. Which registers will R, S and T be?

3. Draw informal pictures to illustrate the following for 32-bit MIPS. **Remember: R_1 and R_2 and R_3 each range over ALL MIPS registers. Can you answer the questions correctly and completely? (At least answer the questions when the registers are all different. There are four other cases in part (b)!)**

   (a) $R_1 := 66$ and then $R_2 := 77$ and then $R_1 := 88$.

   (b) Now continue with $R_3 := !R_1 - !R_2$ and then $R_1 := !R_1 - !R_3$ (all in denary).

   (c) Draw MIPS pictures of $B[5]$, $B[7]$ and $W[12]$. Show 10000001.01111110.10101010.00001000 in $W[12]$ in a MIPS memory. ** Repeat for ARM; choose your own 64-digit number to store.

4. Explain **register direct addressing** by example, drawing some pictures. Illustrate execution of `sub` R_1, R_2, R_3 by taking the registers to be 5-bit, making up some suitable data. You can take the semantics to be $R_1 := !R_2 - !R_3$, where $-$ is 5-bit ALU subtraction.

5. Explain **register indirect addressing** by example, drawing some simple pictures of CPU registers and main memory words/cells as you feel appropriate. Why do we sometimes talk about *pointer* addressing?

6. Explain in detail **indexed addressing** for a MIPS source argument of the form $K(R)$ where $-2^{15} \leq K \leq 2^{15} - 1$. Draw pictures of CPU registers and main memory cells/word locations which help illustrate your answer, taking a denary value for K and the contents of R.