

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 Introductory Digital Systems Laboratory
Problem Set #5

Issued: Wednesday March 13, 2002

Due: Friday March 22, 2002

After numerous lectures, recitations, problem sets, quizzes, and those dreadful labs, you decide you can't take any more of 6.111 and drop it. Seeing as you don't fulfill your laboratory requirement, you do not graduate from college and are forced to take a job at McDonald's to pay off the loans you have accrued from your first few years at MIT. You soon get bored of flipping burgers and seek mental stimulation. You decide to make use of some of that knowledge you gained from 6.111 to build an automated burger stand. McDonald's likes your idea and has already sent out the mechanical system to other departments. You still have to design the control unit though. The following are the inputs they need for their controller, which they dubbed the McControl Unit (McCU):

Sandwich Control Buttons

B1 B2 B3

0	0	0	Big Mac
0	1	0	Quarter Pounder
1	0	0	McChicken Sandwich

Side Order Control Button

F

0	French Fries
1	Onion Rings

Drink Control Button

D

0	Coke
1	7-Up

There is also a button to push ("Process Order") which tells the controller that the order is complete and to start it going. This button is synchronized with the controller.

A customer would select the sandwich he wants by selecting either B0, B1, or B2 switches. Then they select the side order by using the F switch. Finally, they select their drink by using the D switch. When all is correct they press the Process Order button. Assume that the user always selects one sandwich, a side order, and a drink.

The McCU must take these inputs and give proper commands to the Cooker Unit and to the Bagger Unit. The following are the control lines for them.

Cooker Unit Inputs:

LOADBM Load Big Mac
 LOADQP Load Quarter Pounder
 LOADCS Load McChicken Sandwich
 COOK1MIN Cook Sandwich for 1 Minute

Cooker Unit Outputs:

COOKDONE Finished 1 Minute Cook Cycle

Bagger Unit Inputs:

BAGSW Put Sandwich from Cooker Unit into Bag
 BAGOR Put Onion Rings in bag
 BAGFF Put French Fries in bag
 BAGCK Put Cook in bag
 BAG7U Put 7-up in bag
 DISPENSE Dispense the assembled bag of food

The first thing that the McCU must do is to load the correct sandwich into the Cooker Unit. The Cooker Unit has an input (COOK1MIN) which when raised high cooks the sandwich for one minute. When the one minute cook cycle completes the COOKDONE goes high until COOK1MIN goes high again. Each type of sandwich must cook for a different amount of time and it is part of your job to make sure it cooks for the correct amount of time. The Big Mac must cook for 4 minutes, the Quarter Pounder for 3 minutes, and the McChicken Sandwich must cook for 2 minutes. Once the sandwich is finished cooking, you must tell the Bagger Unit to bag the sandwich, then bag the onion rings or french fries. (You don't have to worry about cooking the side orders, because they were all cooked in 2001 and have been sitting under a heat lamp ever since.) Finally, you must bag the correct soda and dispense the food. When all is done you should wait for another order to process.

1. Draw a flow chart of the instruction needed in the McCU. Make sure that the order is processed in the same order as the paragraph above.

2. The following is the instruction set used for the McCU:

If condition	Jump address	0	C2	C1	C0	A4	A3	A2	A1	A0
Assert Signals to Cooker Unit		1	0	S6	S5	S4	S3	S2	S1	S0
Assert Signals to Bagger Unit		1	1	S13	S12	S11	S10	S9	S8	S7

Assign the condition bits C2, C1, and C0 to all of the conditions and make sure you assign one so that you can always do an unconditional jump (make the unconditional jump be C2 C1 C0 = "111"). Assign S6-S0 to be the assertions for the Cooker Unit. You will have three left over so leave S6, S5, S4 unused. Assign S13-S7 to be the assertions to the bagger unit. You will have one left over so leave the S13 bit unused.

3. Design a McCU using two '163 counters, a 9-bit EPROM, on '151 8-input MUX, and one or two 20v8's for assertion logic. Include PAL files for realizing assertion logic.

4. Create a file to be used by the microcode assembler to interpret the microcode that you will write. Complete the file below:

```
/* McCU.sp, spec file for McCU.as */
```

```
/* define op-code and address fields */
```

```
op<8:0>
```

```
/* add your code here */
```

```
/* define op-codes */
```

```
JUMP nop
```

```
/* add your code here */
```

```
/*define condition codes */
```

```
/* add your code here */
```

```
/* define assertions */
```

```
/* add your code here */
```

5. Write the microcode for the McCU. Complete the McCU.as file below:

```
/* McCU.as */
```

```
# SPEC_FILE = McCU.sp; /* Where to find the spec file */
```

```
# LIST_FILE = McCU.lst; /* Where to send listing file */
```

```
# SET_ADDRESS = 0; /* Address to start assembling */
```

```
# START:
```

```
/* Insert your code here. */
```

Turn in a hardcopy of all your code listings.