

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory

Problem Set 5

Issued: March 14, 2001

Due: March 23, 2001

Problem 1: K is for Kompressor

Having lost all of your tuition money gambling on roulette last week, you decide you'd better make it back before the Registrar puts you on reg hold. You manage to pick up a lucrative short-term job building an MCU to control the new line of Kompressor-brand car compactors.

Kompressor's new car compactors is the state-of-the-art in car compactors. It has a conveyor belt, onto which cars are placed, which move the cars into the compactor itself. The compactor smashes cars down to about the size of a metal briefcase before moving it on, and starting on the next one.

The Kompressor MCU takes in four conditional signals, in addition to an always high `TRUE` signal. `IN_KOMPRESSOR` is high when there's a car centered in the Kompressor car compactor, ready for compacting. `NOT_IN_KOMPRESSOR` is the inverse of `IN_KOMPRESSOR`. Note that `IN_KOMPRESSOR` will be true if anything is in the compactor, including an already compacted car, so the MCU will need to check that the compactor is clear (using `NOT_IN_KOMPRESSOR` before checking to see if the next car is ready for crushing. `SIZE_OF_METAL_BRIEFCASE` is true when the compactor has crushed the car down as far as it can, while `SIZE_OF_CAR` is true when the compactor is opened all the way, and the next car can be moved in.

There are three signals the Kompressor MCU needs to generate. The first is the signal to activate the compactor, and the second is a signal telling the compactor whether to compact or release (a directional control). The two MCU assert signals `KOMPRESSOR_COMPACT` and `KOMPRESSOR_RELEASE` are two bits wide, since they need to control both of these signals. The third signal moves the conveyor belt, and since the conveyor belt only moves one direction, `MOVE_CONVEYOR` only needs one bit.

The entire specification file has already been written for you, and the assembly code has been started. Code the rest of the assembly file. If you need to add any other signals, note any changes you would make to the specification file.

The assembly code for the Kompressor is being burned onto a PROM, but the rest of the logic is going to be run by VHDL code. Write the VHDL code for the rest of the system. You may wish to look at slides 17 and 18 of lecture 13 to see how the MCU should be set up. However, note that this is a 4-bit address, 8-bit instruction MCU, so you'll only need one four-bit counter.

Problem 2: Kompressor in Action

The latest Kompressor-brand car compactor has an additional feature. Recently, villains all over the world have been stuffing people into cars that were being sent to car compactors, in order to remove 'problems'. In order to combat villains world-wide, Kompressor has added a victim detector, which causes the compactor to immediately stop compacting the car it's currently working on, and move the car on with the victim still (hopefully) intact on the inside.

Unfortunately, they neglected to mention that until you'd already sent the MCU code out to the factories, and all of the PROMs have been burned (they used the unerasable PROMs, so changing the assembly code isn't going to work). One clever technician suggested connecting the victim detector to a clear signal for the MCU counter, causing the compactor to go to the point where it releases and moves the car along. However, when implemented, if a victim is detected in the compactor, the compactor just stops doing anything. Why is this? How can this problem be fixed, though still using the idea of using the counter's clear? Show the changes you need to make to your VHDL code to implement this.

```

/* kompressor.sp */
/* specification file of the MCU for kompressor-brand car compactors */

op <7:0>;
address op <3:0>;

/* Instruction codes for the MCU:
 * ASSERT indicates that signals should be asserted
 * CJMP indicates a jump to an address if the condition is true
 * JMP indicates a jump to an address regardless of any conditions
 */

ASSERT op<7>=%b1;
CJMP op<7>=%b0;
JMP op<7:4>=%b0111;

/* Conditional signals for use in a CJMP statement. */

KOMPRESSOR_COMPACT op<2:1>=%b11;
KOMPRESSOR_RELEASE op<2:1>=%b10;
MOVE_CONVEYOR op<0>=1;

/* Assertion signals for the ASSERT statements. */

IN_KOMPRESSOR op<6:4>=0;
NOT_IN_KOMPRESSOR op<6:4>=1;
SIZE_OF_METAL_BRIEFCASE op<6:4>=2;
SIZE_OF_CAR op<6:4>=3;
TRUE op<6:4>=7;

/* These are meaningless signals added for readability. */

IF nop;
THEN nop;
KOMPRESSOR_IS nop;

```

```
/* kompressor.as */
/* assembly file of the MCU for kompressor-brand car compactors */

# SPEC_FILE = kompressor.sp;
# LIST_FILE = kompressor.lst;
# MASK_COUNT = 8;
# SET_ADDRESS = 0;
# LOAD_ADDRESS = 0;

/* First, reset the Kompressor car compactor to maximum size. */
RELEASE: IF KOMPRESSOR_IS SIZE_OF_CAR THEN CJMP CONVEYOR;
        ASSERT KOMPRESSOR_RELEASE;
        JMP RELEASE;

/* Next, you add the rest of the code. */
CONVEYOR:
```