University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Checking for Upper-Case Characters

Task: Checking for an Upper-Case Letter

Let's design logic to check whether an **ASCII** character is an upper-case letter.

In **ASCII**, 'A' is **1000001** (0x41), and 'Z' is **1011010** (0x5A).

Let's say that the **ASCII** character is in $\mathbf{C = C_6C_5C_4C_3C_2C_1C_0}$.

How can we check whether C represents an upper-case letter?

We Will Need a BIG Truth Table!

Should we write a truth table for **U(C)**?

Can we skip to the ones that matter?

C_6	C_5	C_4	C_3	C_2	C_1	C_0	U(C)
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	1	0

Let's Break the Truth Table into Eight Pieces

Can we **break the truth table into pieces**?

For example, let's break the truth table
- into eight truth tables
- of 16 rows each.

Each piece represents one value of $\mathbf{C_6C_5C_4}$.

We can solve each piece with a K-map on $\mathbf{C_3C_2C_1C_0}$.

Some Functions are Quite Simple

Or maybe we don't need a K-map for some.

Remember that 'A' is **1000001** (0x41), and 'Z' is **1011010** (0x5A).

Think about the table for $C_6C_5C_4 = 000$?*

What is the function of $C_3C_2C_1C_0$? **0**

(In other words, no **ASCII** character with $C_6C_5C_4 = 000$ is an upper-case letter.)

* This notation means $C_6 = 0$ AND $C_5 = 0$ AND $C_4 = 0$.

Only Two of Our Functions are Not the 0 Function

For reference: 'A' is **1000001** (0x41), and 'Z' is **1011010** (0x5A).

Which of our eight functions are not the 0 function?

$C_6C_5C_4 = 100$ Let's call the function T_4.

$C_6C_5C_4 = 101$ Let's call the function T_5.

Let's solve K-maps for these two.

Solve T_4 Using a Single Loop

Let's solve T_4. **Should we use SOP or POS?**

T_4 is a maxterm!

$T_4 = (C_3 + C_2 + C_1 + C_0)$

T_4 (C_1C_0 \ C_3C_2)	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

Solve T_5 as a POS Expression

Let's solve T_5. POS is better again.

What are the loops? for SOP?

$(C_3' + C_2')$

$(C_3' + C_1' + C_0')$

So $T_5 = (C_3' + C_2') \cdot (C_3' + C_1' + C_0')$

T_5 (C_1C_0 \ C_3C_2)	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	0
10	1	1	0	1

Combine T_4 and T_5 to find U(C)

How do we combine $\mathbf{T_4}$ and $\mathbf{T_5}$ to find the full upper-case checker function **U(C)**?

Remember:
- $\mathbf{T_4}$ applies when $\mathbf{C_6C_5C_4 = 100}$, and
- $\mathbf{T_5}$ applies when $\mathbf{C_6C_5C_4 = 101}$.

So …?
- AND $\mathbf{T_4}$ with $\mathbf{C_6C_5'C_4'}$,
- AND $\mathbf{T_5}$ with $\mathbf{C_6C_5'C_4}$, and
- OR the results together.

A Good Solution, But Maybe We Can Do Less Work?

So $\mathbf{U(C) = C_6C_5'C_4'\,(C_3 + C_2 + C_1 + C_0) + C_6C_5'C_4\,(C_3' + C_2')(C_3' + C_1' + C_0')}$

That's a pretty small and fast solution.

But we still had to do a fair bit of work.

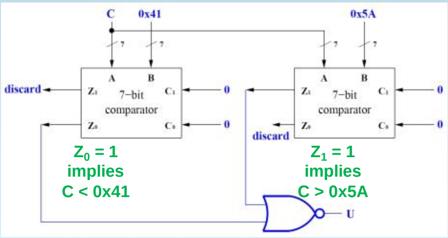
Is there an easier way?

Consider the following: to check for an upper-case letter, we need to know whether

$$\mathbf{C \geq 1000001 \ \ AND \ \ C \leq 1011010}$$

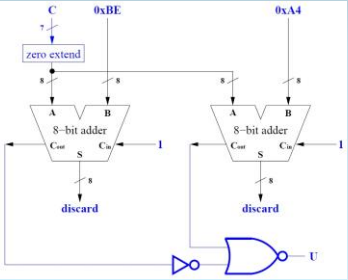
Use Two Comparators to Calculate U(C)

What about this approach?



Or Use Two Adders to Calculate U(C)

Or this approach?



Note that the adders are performing subtractions.

Inefficient, But Simple to Design

Quite large and slow compared with our first solution?

Consider two arguments:

1. CAD tools can optimize away much of the extra overhead.
2. Software executing on data center servers around the world executes the adder design even less efficiently, but it's constantly in use on hundreds of thousands of machines.