Engr 220 Lab

Calvin College Engineering Department

2022

Weekly Lab Schedule

Section A: Wednesday 6:30 – 8:20 p.m.		Section B: Thursday 6:30 – 7:20 p.m.	
Date:	**Topic:**	**Date:**	**Topic:**
Aug 31	Introduction to Simplex, Digital Logic	Sep 1	Introduction to Simplex, Digital Logic
Sep 7	Quartus Design and Simulation, Memory Elements	Sep 8	Quartus Design and Simulation, Memory Elements
Sep 14	State Machine Analysis	Sep 15	State Machine Analysis
Sep 21	Arbiter Design Project	Sep 22	Arbiter Design Project
Sep 28	Intro to NIOS Assembly	Sep 29	Intro to NIOS Assembly
Oct 5	Machine Code	Oct 6	Machine Code
Oct 12	Introduction to the stack	Oct 13	Introduction to the stack
Oct 19	Assembly Function Calls	Oct 20	Assembly Function Calls
Oct 26	Using I/O	Oct 27	Using I/O
Nov 2	Advising Break – No Lab	Nov 3	Using I/O with Interrupts
Nov 9	Using I/O with Interrupts	Nov 10	Using the HAL
Nov 16	Using the HAL	Nov 17	Serial Communication
Nov 23	Thanksgiving Break – No Lab	Nov 24	Thanksgiving Break – No Lab
Nov 30	Serial Communication	Dec 1	Wrap-up and Catch-up
Dec 7	Wrap-up and Catch-up	Dec 8	Exams – No Lab

Last Week

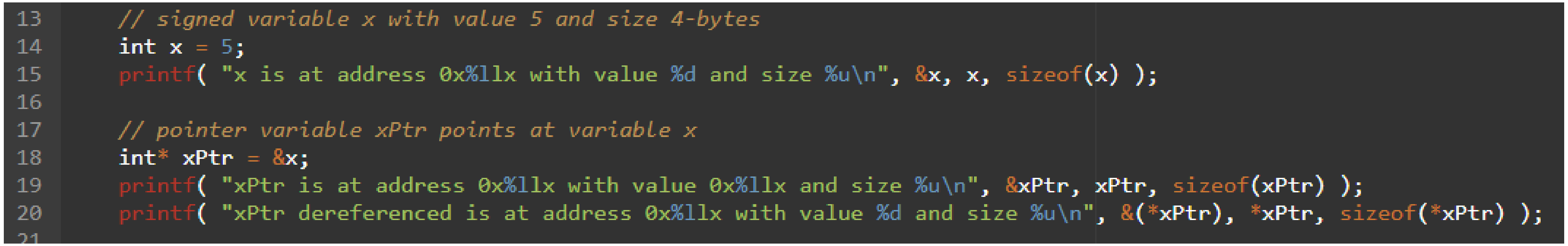
- ASCII Characters & C-Strings
- Serial Communication
- DE2 Board LCD Device
- SignalTap Analyzer

- Questions?

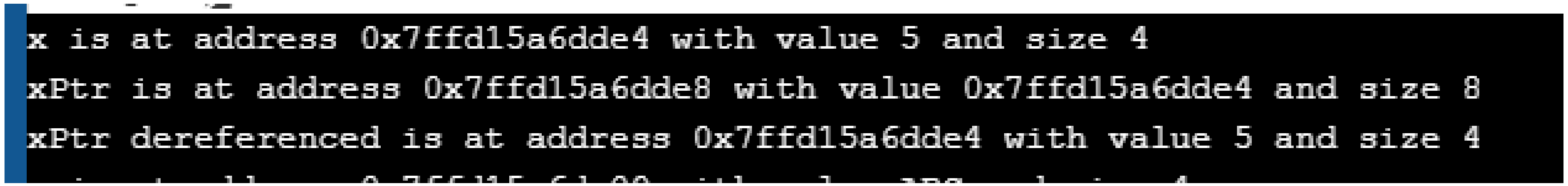
- *No assembly this week*

Pointers in C

- Variables have addresses which can be pointed to (remember Lab10)
- Pointers can be dereferenced



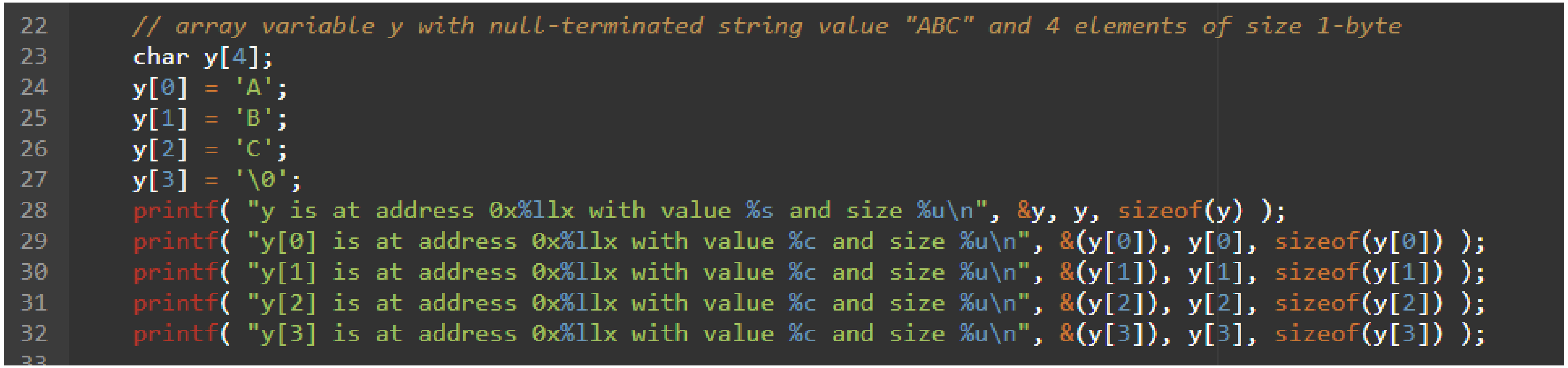
```c
    // signed variable x with value 5 and size 4-bytes
    int x = 5;
    printf( "x is at address 0x%llx with value %d and size %u\n", &x, x, sizeof(x) );

    // pointer variable xPtr points at variable x
    int* xPtr = &x;
    printf( "xPtr is at address 0x%llx with value 0x%llx and size %u\n", &xPtr, xPtr, sizeof(xPtr) );
    printf( "xPtr dereferenced is at address 0x%llx with value %d and size %u\n", &(*xPtr), *xPtr, sizeof(*xPtr) );
```



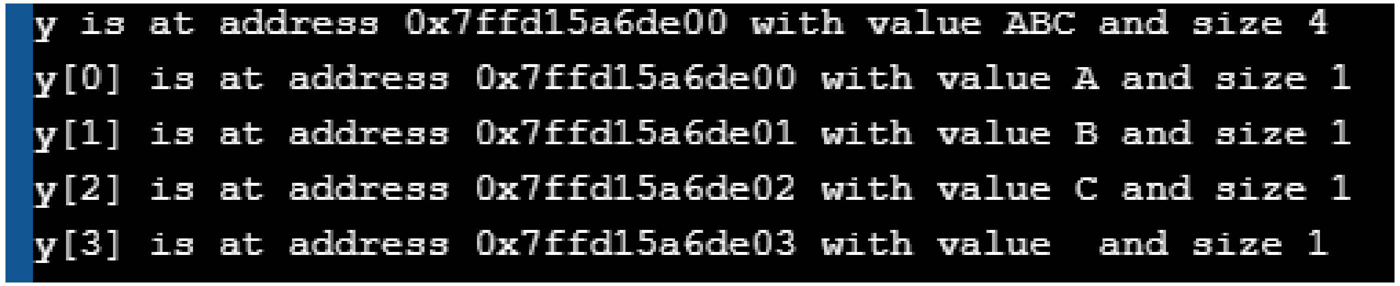
```
x is at address 0x7ffd15a6dde4 with value 5 and size 4
xPtr is at address 0x7ffd15a6dde8 with value 0x7ffd15a6dde4 and size 8
xPtr dereferenced is at address 0x7ffd15a6dde4 with value 5 and size 4
```

Pointers in C

- Arrays are similar to pointers to the first element (remember Lab 8)



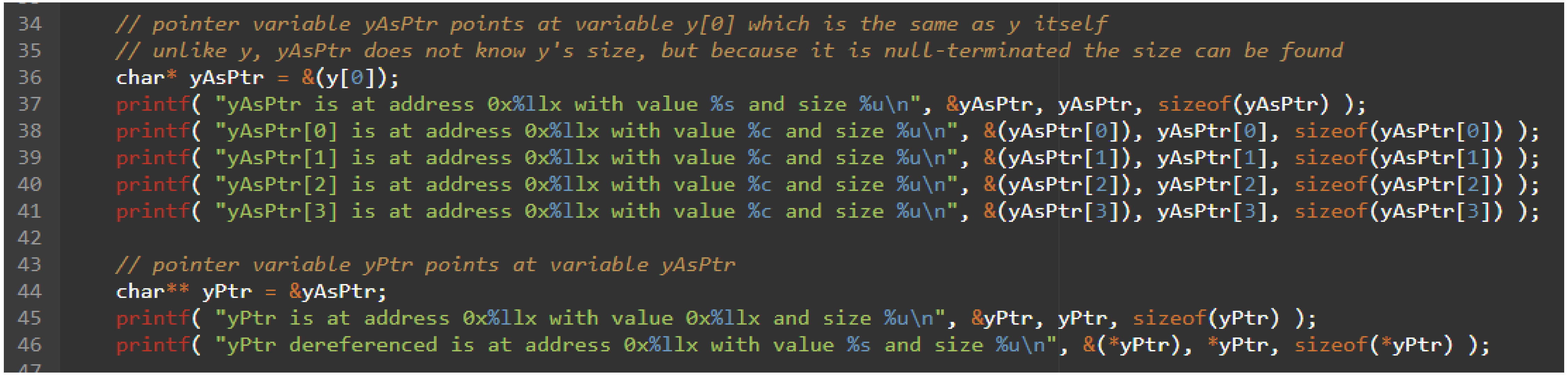
```c
    // array variable y with null-terminated string value "ABC" and 4 elements of size 1-byte
    char y[4];
    y[0] = 'A';
    y[1] = 'B';
    y[2] = 'C';
    y[3] = '\0';
    printf( "y is at address 0x%llx with value %s and size %u\n", &y, y, sizeof(y) );
    printf( "y[0] is at address 0x%llx with value %c and size %u\n", &(y[0]), y[0], sizeof(y[0]) );
    printf( "y[1] is at address 0x%llx with value %c and size %u\n", &(y[1]), y[1], sizeof(y[1]) );
    printf( "y[2] is at address 0x%llx with value %c and size %u\n", &(y[2]), y[2], sizeof(y[2]) );
    printf( "y[3] is at address 0x%llx with value %c and size %u\n", &(y[3]), y[3], sizeof(y[3]) );
```



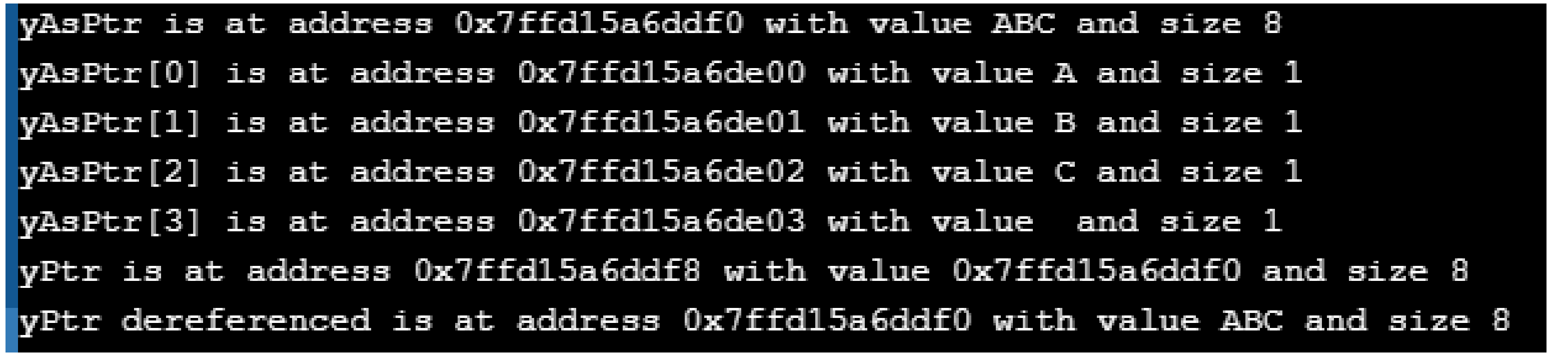
```
y is at address 0x7ffd15a6de00 with value ABC and size 4
y[0] is at address 0x7ffd15a6de00 with value A and size 1
y[1] is at address 0x7ffd15a6de01 with value B and size 1
y[2] is at address 0x7ffd15a6de02 with value C and size 1
y[3] is at address 0x7ffd15a6de03 with value  and size 1
```

Pointers in C

- char* is similar to char[], both can be used as strings



```c
    // pointer variable yAsPtr points at variable y[0] which is the same as y itself
    // unlike y, yAsPtr does not know y's size, but because it is null-terminated the size can be found
    char* yAsPtr = &(y[0]);
    printf( "yAsPtr is at address 0x%llx with value %s and size %u\n", &yAsPtr, yAsPtr, sizeof(yAsPtr) );
    printf( "yAsPtr[0] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[0]), yAsPtr[0], sizeof(yAsPtr[0]) );
    printf( "yAsPtr[1] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[1]), yAsPtr[1], sizeof(yAsPtr[1]) );
    printf( "yAsPtr[2] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[2]), yAsPtr[2], sizeof(yAsPtr[2]) );
    printf( "yAsPtr[3] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[3]), yAsPtr[3], sizeof(yAsPtr[3]) );

    // pointer variable yPtr points at variable yAsPtr
    char** yPtr = &yAsPtr;
    printf( "yPtr is at address 0x%llx with value 0x%llx and size %u\n", &yPtr, yPtr, sizeof(yPtr) );
    printf( "yPtr dereferenced is at address 0x%llx with value %s and size %u\n", &(*yPtr), *yPtr, sizeof(*yPtr) );
```



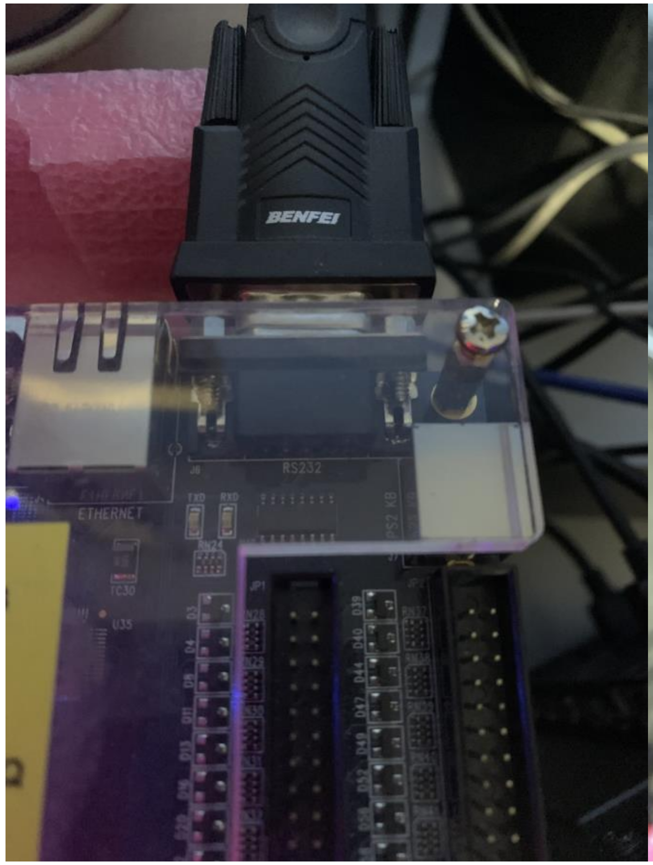
```
yAsPtr is at address 0x7ffd15a6ddf0 with value ABC and size 8
yAsPtr[0] is at address 0x7ffd15a6de00 with value A and size 1
yAsPtr[1] is at address 0x7ffd15a6de01 with value B and size 1
yAsPtr[2] is at address 0x7ffd15a6de02 with value C and size 1
yAsPtr[3] is at address 0x7ffd15a6de03 with value  and size 1
yPtr is at address 0x7ffd15a6ddf8 with value 0x7ffd15a6ddf0 and size 8
yPtr dereferenced is at address 0x7ffd15a6ddf0 with value ABC and size 8
```

Pointers in C

- Online Tutorials
  - https://www.tutorialspoint.com/cprogramming/c_pointers.htm
  - http://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/
  - https://www.programiz.com/c-programming/c-pointers
  - https://www.cprogramming.com/tutorial/c/lesson6.html

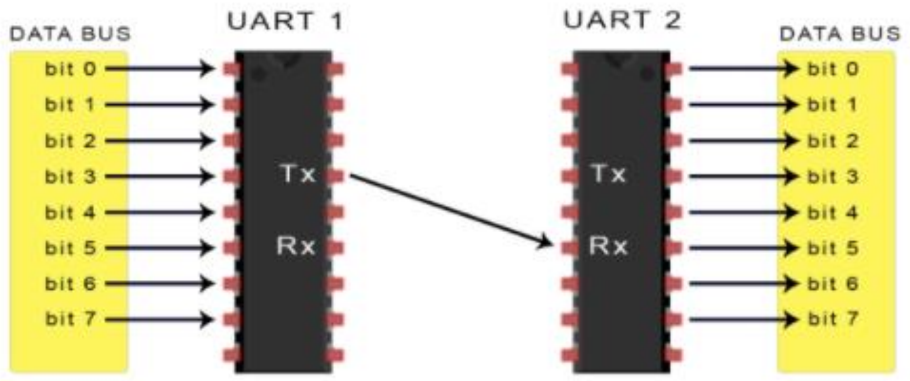
DE2 UART Device

- Another I/O device which is capable of sending and receiving bytes of data
- 9 pins, 7 data, RX, and TX
  - https://www.circuitbasics.com/basics-uart-communication/
  - https://en.wikipedia.org/wiki/RS-232

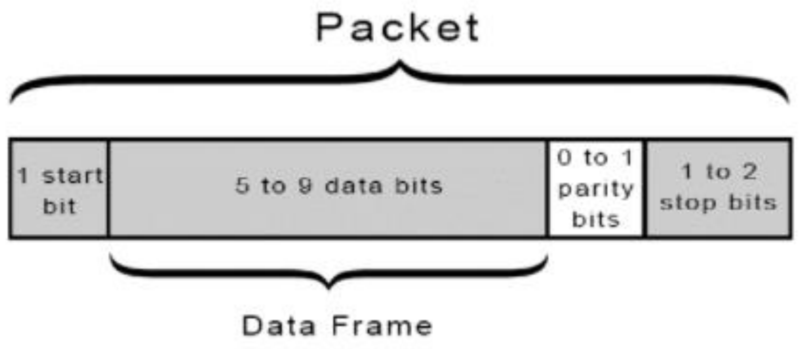


DE2 UART Device

- Another I/O device which is capable of sending and receiving bytes of data
- 9 pins, 7 data, RX, and TX
  - https://www.circuitbasics.com/basics-uart-communication/
  - https://en.wikipedia.org/wiki/RS-232

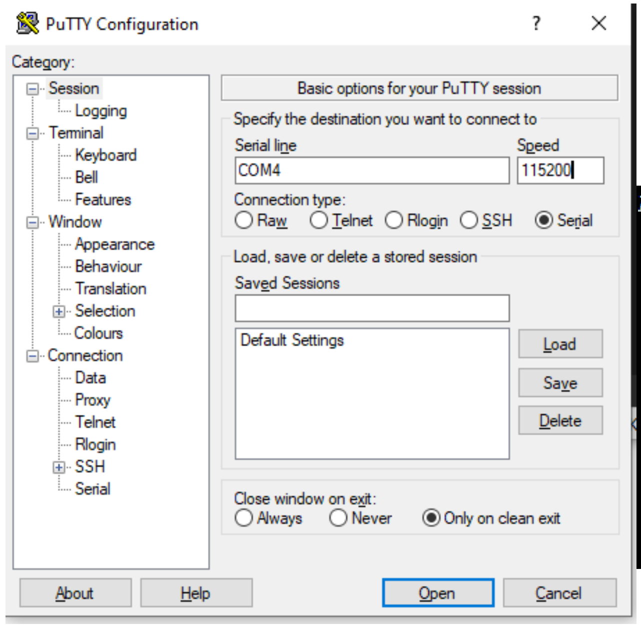


UART transmitted data is organized into *packets*. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional *parity* bit, and 1 or 2 stop bits:



DE2 UART Device

- Another I/O device which is capable of sending and receiving bytes of data
- 9 pins, 7 data, RX, and TX
- https://www.circuitbasics.com/basics-uart-communication/
- https://en.wikipedia.org/wiki/RS-232



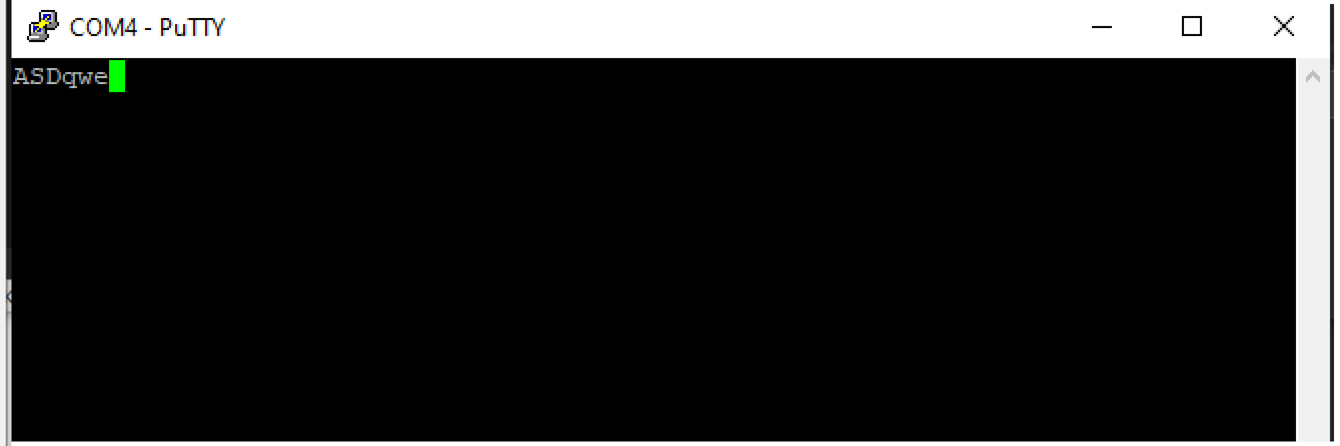
Specify the destination you want to connect to

Serial line: COM4

Speed: 115200

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin ( ) SSH [✓ Serial]

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]



```
ASDqwe
```

UART Serial Communication

- DE2 Board UART Device Register Bit-Fields
- 7-bits of ASCII data provided in time
- RAVAIL shows quantity of data remaining, decreases to 0

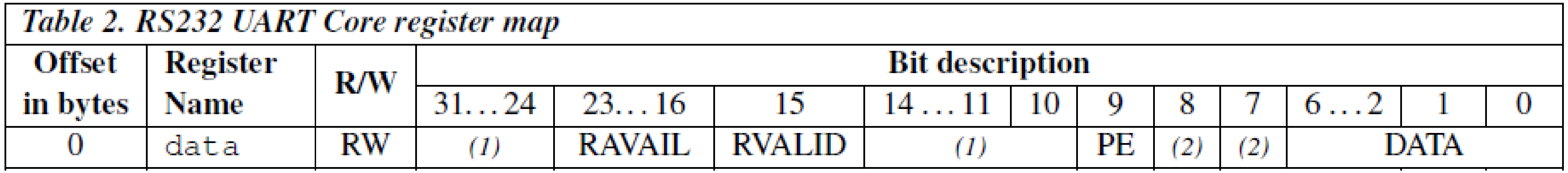


Table 2. RS232 UART Core register map

Offset in bytes	Register Name	R/W	Bit description										
			31…24	23…16	15	14…11	10	9	8	7	6…2	1	0
0	`data`	RW	*(1)*	RAVAIL	RVALID	*(1)*		PE	*(2)*	*(2)*	DATA		

UART Serial Communication

- DE2 Board UART Device Register Bit-Fields
- 7-bits of ASCII data provided in time
- RAVAIL shows quantity of data remaining, decreases to 0
- Implicit data queue (FIFO) is hidden from us, RAVAIL is its size
  - What might happen if we publish to queue faster than we consume from it?

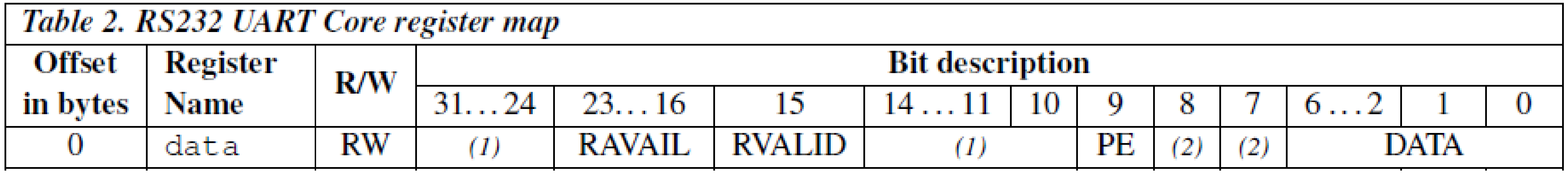
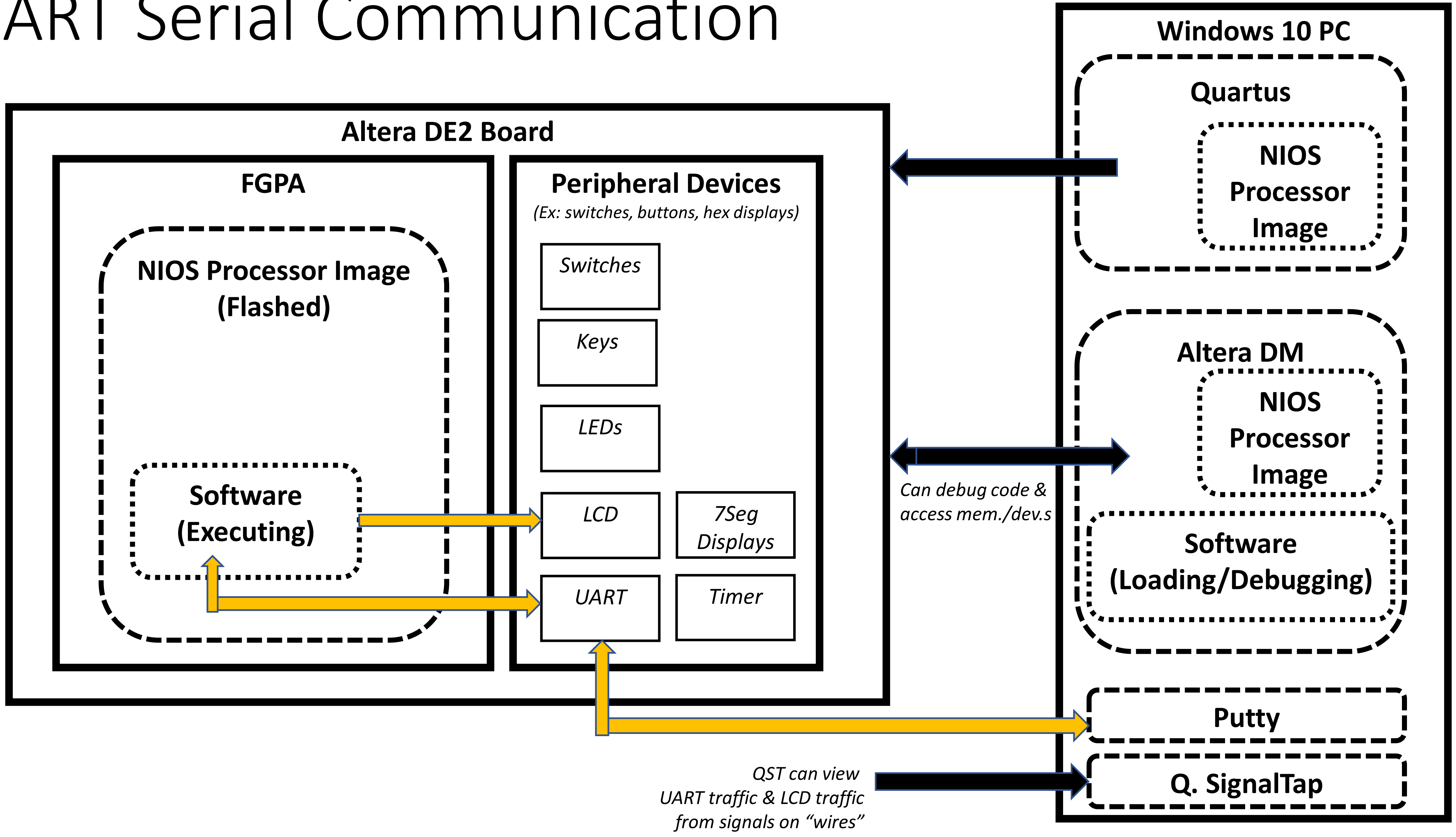


Table 2. RS232 UART Core register map

Offset in bytes	Register Name	R/W	Bit description										
			31…24	23…16	15	14…11	10	9	8	7	6…2	1	0
0	`data`	RW	*(1)*	RAVAIL	RVALID	*(1)*		PE	*(2)*	*(2)*	DATA		

UART Serial Communication


Altera DE2 Board
FGPA
NIOS Processor Image (Flashed)
Software (Executing)
Peripheral Devices
(Ex: switches, buttons, hex displays)
Switches
Keys
LEDs
LCD
7Seg Displays
UART
Timer
Windows 10 PC
Quartus
NIOS Processor Image
Altera DM
NIOS Processor Image
Software (Loading/Debugging)
Putty
Q. SignalTap
Can debug code & access mem./dev.s
QST can view UART traffic & LCD traffic from signals on "wires"

DE2 UART Device

- Show reference examples
  - https://calvincollege-my.sharepoint.com/:v:/g/personal/ajj6_calvin_edu/Ec8zCBIvzV9JhnlDqSoTUOYB6hY3tL6xGzq6SKRyXgZdPg?e=mAnhyd
  - https://calvincollege-my.sharepoint.com/:v:/g/personal/ajj6_calvin_edu/EYrKXugFCtRGg6XEoi1rj6gBdhFvdQHWlWG2H98QmUrXsQ?e=j6iMpO

(Links from 2020)

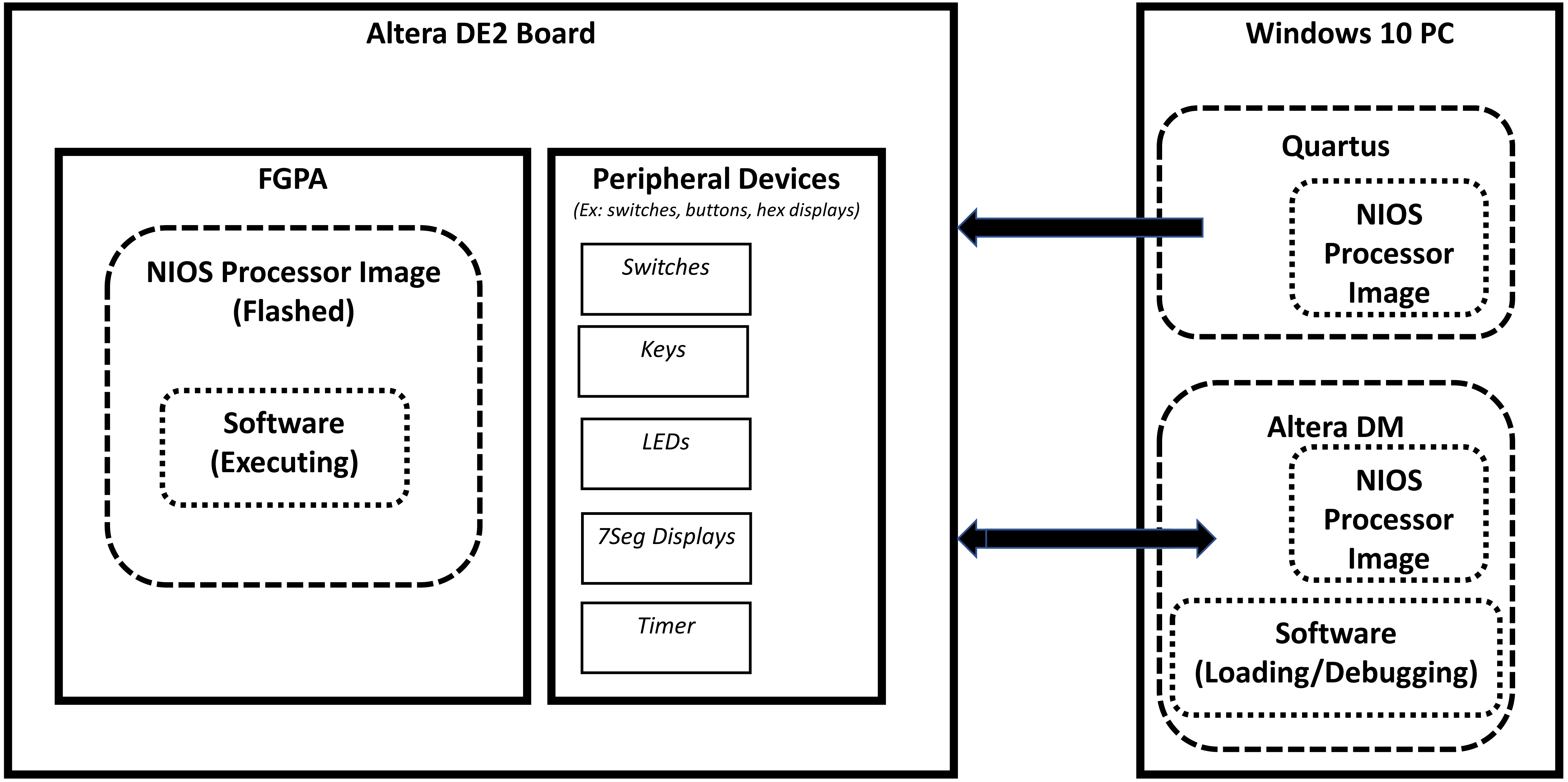
Lab 12 Tips

- Refer to the LCD reference & example therein
- Refer to the UART reference

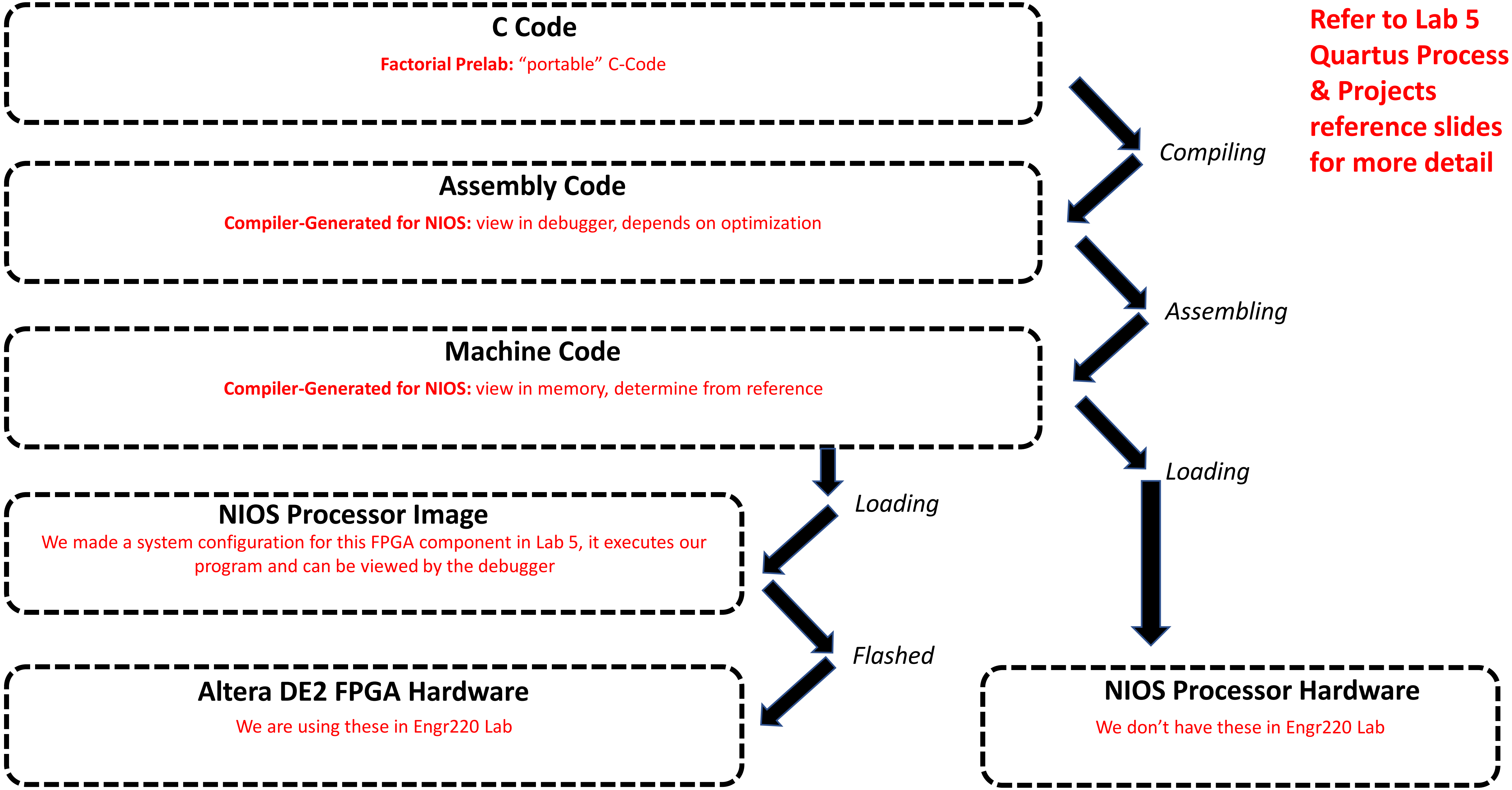
- Take a minute to review Lab 11, Lab 12 is based on Lab 11
- Refer to the Lab 11 slide deck for examples of ASCII characters & strings

- Use “Program with Device Driver Support” to get the BSP

Lab Components



Refer to Lab 5 Quartus Process & Projects reference slides for more detail



C Code

Factorial Prelab: "portable" C-Code

Assembly Code

Compiler-Generated for NIOS: view in debugger, depends on optimization

Machine Code

Compiler-Generated for NIOS: view in memory, determine from reference

NIOS Processor Image

We made a system configuration for this FPGA component in Lab 5, it executes our program and can be viewed by the debugger

Altera DE2 FPGA Hardware

We are using these in Engr220 Lab

NIOS Processor Hardware

We don't have these in Engr220 Lab

Char and String Example

```c
/******************************************************************************
                            Online C Compiler.
                Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.
*******************************************************************************/
#include <stdio.h>
#include <string.h>
int main()
{
    // We looked at printf codes before, %c is for characters %s is for strings
    printf( "We will show some examples of characters like '%c' "
            "and strings like \"%s\"\n", 'A', "Hi" );

    // Here is a character
    char letterY = 'Y';
    printf( "letterY: %c\n", letterY );

    // We can do math on characters, because they are just numbers
    char letterZ = ( letterY + 1 );
    printf( "letterZ: %c\n", letterZ );

    // char* or char[] is an array of characters, meaning a string
    char* stringABC = "ABC";
    printf( "stringABC: %s\n", stringABC );

    // strings must be null-terminated (\0 not shown when hard-coded)
    char stringXYZ[4];
    stringXYZ[0] = 'X';
    stringXYZ[1] = 'Y';
    stringXYZ[2] = 'Z';
    stringXYZ[3] = '\0';
    printf( "stringXYZ: %s\n", stringXYZ );

    // string lengths do not usually include the null-terminating char
    printf( "stringXYZ is %u in length\n", strlen(stringXYZ) );

    return 0;
}
```

Copy/Paste text from powerpoint into text editor to view

Pointers Example

```
/******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
Code, is at address 0x%llx with value 0x%llx and size %u\n", &xPtr, xPtr, sizeof(xPtr) );
    printf( "xPtr dereferenced is at address 0x%llx with value %d and size %u\n", &(*xPtr), *xPtr, sizeof(*xPtr) );

    // array variable y with null-terminated string value "ABC" and 4 elements of size 1-byte
    char y[4];
    y[0] = 'A';
    y[1] = 'B';
    y[2] = 'C';
    y[3] = '\0';
    printf( "y is at address 0x%llx with value %s and size %u\n", &y, y, sizeof(y) );
    printf( "y[0] is at address 0x%llx witCompile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>

int main()
{
    // signed variable x with value 5 and size 4-bytes
    int x = 5;
    printf( "x is at address 0x%llx with value %d and size %u\n", &x, x, sizeof(x) );

    // pointer variable xPtr points at variable x
    int* xPtr = &x;
    printf( "xPtr h value %c and size %u\n", &(y[0]), y[0], sizeof(y[0]) );
    printf( "y[1] is at address 0x%llx with value %c and size %u\n", &(y[1]), y[1], sizeof(y[1]) );
    printf( "y[2] is at address 0x%llx with value %c and size %u\n", &(y[2]), y[2], sizeof(y[2]) );
    printf( "y[3] is at address 0x%llx with value %c and size %u\n", &(y[3]), y[3], sizeof(y[3]) );

    // pointer variable yAsPtr points at variable y[0] which is the same as y itself
    // unlike y, yAsPtr does not know y's size, but because it is null-terminated the size can be found
    char* yAsPtr = &(y[0]);
    printf( "yAsPtr is at address 0x%llx with value %s and size %u\n", &yAsPtr, yAsPtr, sizeof(yAsPtr) );
    printf( "yAsPtr[0] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[0]), yAsPtr[0], sizeof(yAsPtr[0]) );
    printf( "yAsPtr[1] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[1]), yAsPtr[1], sizeof(yAsPtr[1]) );
    printf( "yAsPtr[2] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[2]), yAsPtr[2], sizeof(yAsPtr[2]) );
    printf( "yAsPtr[3] is at address 0x%llx with value %c and size %u\n", &(yAsPtr[3]), yAsPtr[3], sizeof(yAsPtr[3]) );

    // pointer variable yPtr points at variable yAsPtr
    char** yPtr = &yAsPtr;
    printf( "yPtr is at address 0x%llx with value 0x%llx and size %u\n", &yPtr, yPtr, sizeof(yPtr) );
    printf( "yPtr dereferenced is at address 0x%llx with value %s and size %u\n", &(*yPtr), *yPtr, sizeof(*yPtr) );

    return 0;
}
```

Copy/Paste text from powerpoint into text editor to view