Lab Lecture 2 w/ Rave ♪

Thursday, February 4, 2021 1:17 PM

1. Lab Announcements
2. Lab 2 Requirements
3. ASM v. C
   - Variables
   - Arrays
   - Conditional statements
   - Loops
4. Function Calls
5. Debugging in C

① Lab Announcements

↳ Grading Sheets https://drive.google.com/drive/folders/0B-DcTAx1HVlydWw4UmxycmtHc1E

↳ Standardized grading
- −5 pts per question not answered completely in Demonstration

* Any issues w/ grading contact TA who checked your lab out

↳ Late lab checkout

Date of your lab / Date of your Turn in	Tuesday	Wednesday	Thursday	Friday	Monday	Tuesday
Tuesday	0/-5 points	-10 points	-20 points	-30 points	-45 points	Not allowed
Wednesday		0/-5 points	-10 points	-20 points	-35 points	Not allowed
Thursday			0/-5 points	-10 points	-25 points	-40%

↳ Labs 3-10 Partner Based
* look out for spreadsheet in coming days
* start looking for partners
  ↳ if having trouble contact any TA

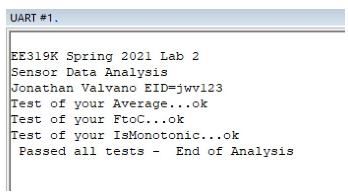
* PARTNERS MUST BE IN YOUR SECTION
* SIGN UP FOR 1 TIME SLOT IN YOUR SECTION

② Lab 2 Requirements

↳ Write 3 C Functions
- Average
- FtoC
- isMonotonic

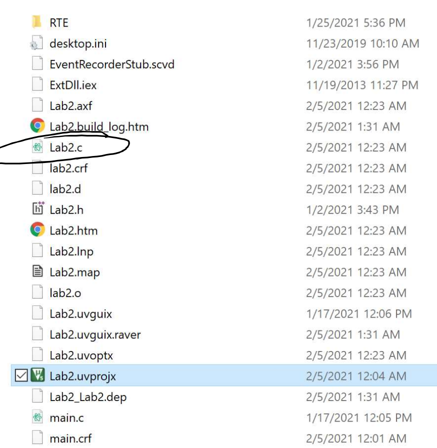
↳ Run through autograder (UART #1 window "All tests passed")

↳ Deliverables — Lab2.c



* if ever lose UART window, go to Serial windows → UART #1 in debugging mode

Functions Breakdown



Average
computing avg from Data array of N elements

FtoC
Eqn: $C = (F-32)\frac{5}{9}$

* integer division ($\frac{5}{9}$ evaluates to 0,) separate operations

isMonotonic
checking for strictly increasing or equal #s

③ ASM v. C

A. Variables

(RAM)

ASM		C
fun SPACE 1	8-bit = byte	(uint8_t fun;
nun SPACE 2	16-bit = halfword	uint16_t nun;
done SPACE 4	32-bits = word	int32_t done; (int)

(label SPACE # of bytes) * 1 byte = 8 bits

(ROM)

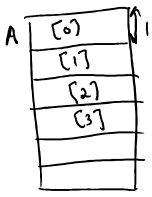
ASM	C
foo DCB 2 (8 bit)	const int8_t foo = 2;
DCW (16 bit)	const int16_t foo = 2;
DCD (32 bit)	const int16_t foo = 2;

B. Arrays (consecutive bytes of memory that hold values)

(RAM) ASM C

uint8_t array[6] = {0};

B. Arrays (consecutive bytes of memory that hold values)



(RAM) ASM

(label SPACE # of bytes)

A SPACE 4 ← name

↳ array 4 1 byte elements
2 2 byte elements
1 4 byte element

(ROM)

delay DCD 2000000, 4000000, 6000000
↑ 32 bit entries

C

uint8_t array[6];
uint16_t foo[3] = {0, 1, 2};

uint8_t array[6] = {0};

const int32_t delay[2] = {2000000, 4000000};

C. Conditional statements

ASM

- look at condition codes
- branch accordingly

BEQ
BNE

signed	unsigned
BLT	BLO
BLE	BLS
BGT	BHI
BGE	BHS

ex.
ADDS R0, R0, R1
BGT label

ex.
CMP R1, #7
BEQ done

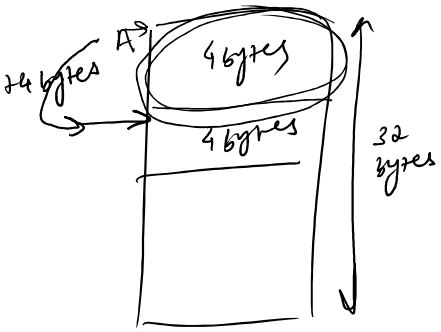
C

if (condition is true) {
execute
}

ex. if (day == Friday) {
rave();
}else{
grind();
}

D. Iterating (loops)

ex. We have 8 32 bit #s (4 bytes each) for which we will add 1 to all of them.



ASM

```
A SPACE 32 bytes ; array of 32 bit #s
B DCB 8          ; # of elements in array

LDR R0, =A
LDR R1, =B       ; R1 = address of B
LDRB R1, [R1]    ; R1 = 8
loop CMP R1, #0  ; check if end of array
BEQ done
LDR R2, [R0]
ADD R2, #1
STR R2, [R0]
ADD R0, #4       ; get to next element in array
                 ; 32 bits = 4 bytes
SUB R1, #1
B loop
done BX LR
```

C

for loops

FORMAT:

ex. for (int8_t i=0; i<8; i++){
arr[i] = arr[i]+1;

for loops

FORMAT:

```
for(initialization; condition; condition update){
        execute
}
```

ex.
```
for(int8_t i=0; i<8; i++){
    arr[i] = arr[i]+1;
    // arr[i] += 1;
    // arr[i]++;
}
```

while loops

FORMAT:

```
while(condition is true){
        execute
}
```

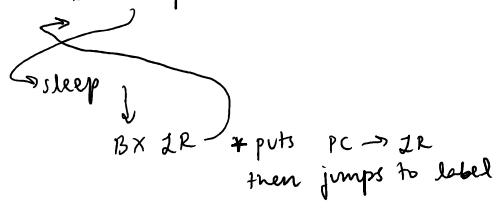
ex.
```
int count = 0;
while(count < 8){
    arr[count] += 1;
    count++;
}
```

(4) Function Calls

ASM

- subroutines
- BL (branch w/ link)
  use w/ subroutine
- BX LR
  return from subroutine

ex. BL sleep



C

- functions
- defined prototype

```
int32_t myFun(int32_t record);
```

int32_t: return parameter; myFun: name; int32_t record: input parameter

ex.
```
int16_t Add(int8_t num1, int8_t num2); }
int16_t sum;
sum = Add(3, 4);
```

num1 will get 3
num2 will get 4

ORDER MATTERS

(5) Debugging in C

- Use breakpoints and step through
- Use watch windows to watch variables while debugging
  - ↳ Right click on variable in code and Add to watch 1

Watch 1

Name	Value	Type
Data[i]	5	ushort
i	0	int
Data[i+1]	0x000C	ushort
<Enter expres...		