Lab 2

Saturday, February 6, 2021 8:54 PM

Start @ 6:33

The Plan

1. General Announcements
2. Lab 2 Overview
3. ASM vs. C
   - ↳ var
   - ↳ array
   - ↳ conditionals
   - ↳ loops
4. Functions

1

- lab grading
  - 2-3 q's
    - if partial or full answer
      - → full credit
      - ↳ will get guidance
    - no understanding, even w/ hint ⇒ -5 pts

- if you have a problem w/ your lab grade, contact your checkout TA.

- Late Policy

Submit day → canvas + checkoff

Your lab day	Tu	W	Th	F	M	Tu
Tue	-0 / -5	-10	-20	-30	-45	0 / 100
Wed		0 / -5	-10	-20	-35	0 / 100
Thur			0 / -5	-10	-25	-40

- Partners
  - ↳ 3-10 labs
  - ↳ same section

Contact TAs if you're stuck

- ↳ new sign up sheet out soon
  - 1 slot
  - w/in section

---

(2)

Goal: write 3 fxns in C

Outcome: Run autograder
- ↳ Build (In keil)
- ↳ Debug Session
- ↳ Run
- ↳ Result in UART #1 window → serial windows → UART
  - 100 == "Passed all tests"

write code in Lab2.c, NOT main.c

Fxn 1) Average (O) of elements in an array (I) of size N. (I)

Fxn 2) convert temp (I) to Celcius (O)

- $C = \frac{5}{9}(F-32)$
- $\frac{5}{9} = 0$ bc int division

$-\frac{5}{9} = 0$ bc int division

Fxn 3) isMonotonic

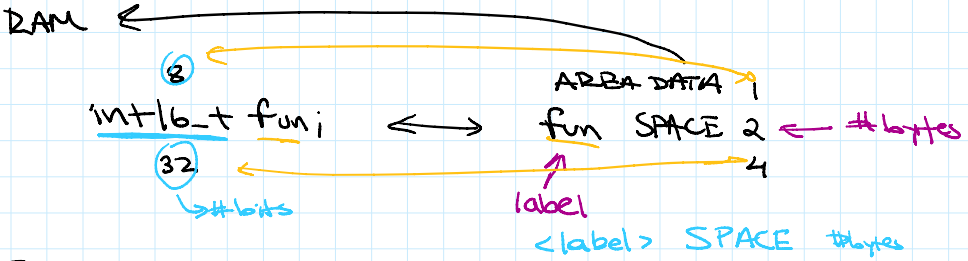
check for strictly increasing or equal

T ← 0 0 1 2 3 100
T ← 0 0 0 0 0
F ← 0 -1 -2 0

---

(3)

Variables



ROM

const int32_t foo = 2; ↔ AREA CODE
foo DCD 2
dtype label = value; <label> DC__ #bits

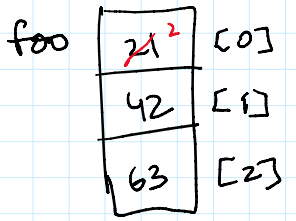
DCD – 32 bit 4 bytes
→ DCW – 16 bit 2 bytes 16 bit = "word"
→ DCB – 8 bit 1 byte

Arrays

C

RAM
uint8_t arr[6];
uint16_t foo[3] = {21, 42, 63};

datatype label[n];



foo[0] = 21;
foo[0] = 2;
uint16_t firstel = foo[2];
↳ 2

ROM const int32_t arr[2] = {1000, 3000};

ROM [const int32_t arr[2] = {1000, 3000};]

arr[0] <- 2;

ASM

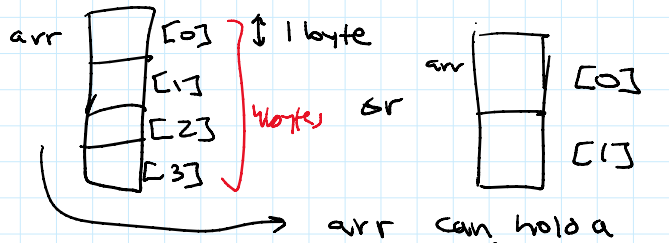
RAM

↳ use SPACE

bytes

arr SPACE 4

label SPACE #b



arr can hold a total of 4 bytes

ROM

↳ use DC_

arr DCD 1000, 2000

↓

32 bits

Conditionals

C:

```
if (condition) {
        // execute if
        // condition is
        // true
} else {
        // exec if cond
        // is false
}
```

ex.

```
if (today == M) {
        // go to lab lec
} else {
        // do your lab
}
```

```
if ( something ) {
        // do something
}
```

```
if ( ) {
```

in ASM use condition codes

in ASM use condition codes carefully

- BEQ label ; BRANCH EQUAL
- BNE label ; BRANCH NOT EQUAL
- BLE label ; BRANCH LESS THAN EQUAL
- BLT label ; BRANCH LESS THAN
- BGE label ; BRANCH GREATER THAN EQUAL
- BGT label ; BRANCH GREATER THAN

signed

un-signed

- BLO <=
- BLS <
- BHI >=
- BHS >

```
if ( ){
    s
} else if ( ){
    M
} else if ( ){
    L
} else {
    XL
}
```

ex: CMP R1, #0x10

BEQ done if

else{

}

done //statements

? R1 == #0x10

BEQ

Loops

for loop

```
for( init ; condition; update){
    // statement(s)
}
```

```
for(uint8_t i=0; i<N ; i++){
    // print( arr[i])
}
```

while loop

initialization

```
while ( condition){
    // statements(s)
    update w/in the codeblock
}
```

conversion ex:

C

```
int i=0;
while (i<12) {
    // stuff
    i++;
}
```

ASM

```
    MOV  R0, #0
loop CMP  R0, #12
    BGE  exit
    // stuff
    ADD  R0, #1
    B    loop
exit BX   LR
```

R0 ___ 12

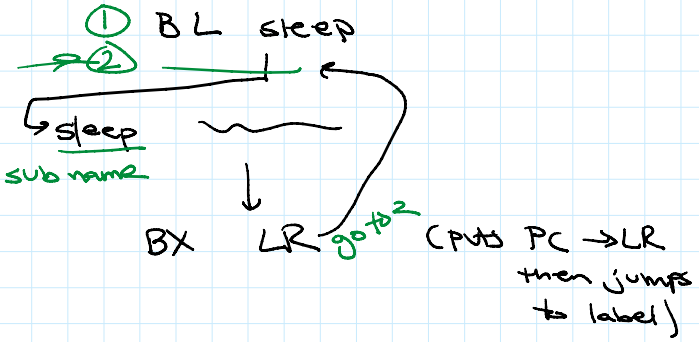
---

(4)

ASM

"subroutines"

- BL (branch w/ link)
- BX LR ("return")



"fxns"

ret type → int8_t

fxnname → LabLec2(int16_t content, int second)

input param (x1) (order matters)

← prototype

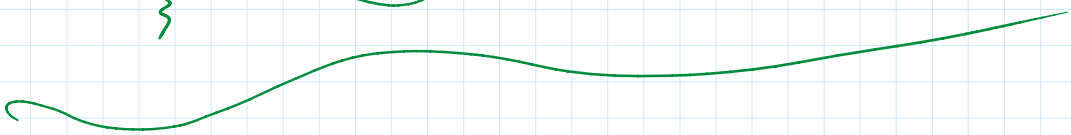
(int content, int second Lablec)

0

```
int8_t hi = LabLec2(8, 1);
```

```
int8_t    LL2(int16_t content);

int8_t    LL2(int16_t content){
    // do stuff
    return 1;
}
```



```
do {
  // 2
} while()
```