EE319K Lecture Lec09.ppt in class worksheet

Question 1. What does it mean for a variable to have private scope?

Question 2. What does it mean for a variable to have public scope?

Question 3. What are two ways to implement dynamic allocation of variables?

Question 4. This function will have 3 local variables on the stack. Each is 32-bit signed.
Part a) Show the binding naming the variables x,y,z (using SP addressing)
Part b) Show the allocation in assembly language
Part c) Show assembly code that sets x=6
Part d) Show assembly code that sets y=1.5*x (1.5 is 1 + ½)
Part e) Show the assembly code that deallocates the variables

Question 5. This function will have 3 local variables on the stack. Each is 32-bit signed.
Part a) Show the binding naming the variables x,y,z (using R11 addressing)
Part b) Show the allocation in assembly language
Part c) Show assembly code that sets x=6
Part d) Show assembly code that sets y=1.5*x (1.5 is 1 + ½)
Part e) Show the assembly code that deallocates the variables

Question 6. We need to store values from -100 to 100 cm with a resolution of at least 0.01cm. If there will be a lot of human input/output and not much internal calculations, what format would you use? Select the fixed constant and the smallest integer type.

Question 7. We need to store values from -100 to 100 cm with a resolution of at least 0.01cm. If there will be a lot of internal calculations and not much human input/output, what format would you use? Select the fixed constant and the smallest integer type.

Question 8. Write a function in C, which uses fixed point calculations, to calculate the area of a circle. The input parameter is radius in mm, and the output is area in mm^2.

Question 9a. List the meaning of these pins that connect the microcontroller to the LCD:

```
// pin 8  SCK          PA2 (SSI0Clk)
// pin 7  MOSI         PA5 (SSI0Tx)
// pin 6  TFT_CS       PA3 (SSI0Fss)
// pin 4  D/C          PA6 (GPIO)
// pin 3  RESET        PA7 (GPIO)
```

Question 9b. List the meaning of these pins that connect the microcontroller to the SSD1306 OLED:

```
// SCL   PB2 I2C clock
// SDA   PB3 I2C data
```

Question 10a. There are four steps to output data to the LCD. Explain the purpose of each step.

```
;1) Read SSI0_SR_R and check bit 1,
;2) If bit 1 is low loop back to step 1
;3) Set D/C=PA6 to one
```

```
;4) Write the 8-bit data to SSI0_DR_R
```

Question 10b. These are the first are four steps to output data to the SSD1306 OLED. Explain the purpose of each step.

```
; 1) wait while I2C is busy, wait for I2C3_MCS_R bit 0 to be 0
; 2) write slave address to I2C3_MSA_R,
;     MSA bits7-1 is slave address
;     MSA bit 0 is 0 for send data
; 3) write first data to I2C3_MDR_R
; 4) write 0x03 to I2C3_MCS_R,  send no stop, generate start, enable
```

Question 11. Explain how the LCD/OLED device driver described in Lab 7 represents an abstraction.

Question 13. What does the assembly pseudo-op IMPORT do?

Question 14. What does the assembly pseudo-op EXPORT do?

Question 15. Consider an 8-bit unsigned integer used with decimal fixed point that has a resolution of 10^-3. What range of values can be represented?

Question 16. Consider an 8-bit signed integer used with binary fixed point that has a resolution of 2^-3. What range of values can be represented?

The following question is interesting, but not on the Spring 2021 Quiz8

Question 12. This function will have a local array on the stack. The array is 100 16-bit numbers. Place local variable **i** in a register.

```
void Example(void){ uint16_t buf[100];
  buf[6] = 6;
  for(int i=0; i<100; i++){
    buf[i] = i;
  }
}
```

Part a) Show the binding naming the array **buf** (using SP addressing)
Part b) Show the allocation in assembly language
Part c) Show assembly code that sets **buf[6]=6**
Part d) Show assembly code that sets **buf[i]=i**
Part e) Show the assembly code that deallocates the array

Answer 1. Private scope means there is a restriction to which programs can access the variable. Examples include restrict to functions in the same file, place within the same function, or places within the same block.
Answer 2. Public scope means any software within the project can access the variable. To access the variable from another file, that file needs to add an **`extern`** definition.
Answer 3. Implement dynamic allocation using registers or on the stack.
Answer 4. This function will have 3 local variables on the stack. Each is 32-bit signed.
Part a) Show the binding naming the variables x,y,z (using SP addressing)

```
x     EQU  0  ;32-bit signed number
y     EQU  4  ;32-bit signed number
z     EQU  8  ;32-bit signed number
```

Part b) Show the allocation in assembly language

```
Func SUB  SP,#12     ;allocate x,y,z
```

Part c) Show assembly code that sets x=6

```
     MOV  R0,#6
     STR  R0,[SP,#x] ;x=6
```

Part d) Show assembly code that sets y=1.5*x (1.5 is 1 + ½)

```
     LDR  R0,[SP,#x] ;R0=x
     MOV  R1,R0
     ASR  R1,R1,#1   ;0.5x
     ADD  R0,R0,R1   ;1.5x
     STR  R0,[SP,#y]
```

Part e) Show the assembly code that deallocates the variables

```
     ADD  SP,#12     ;deallocation
     BX   LR
```

Answer 5. This function will have 3 local variables on the stack. Each is 32-bit signed.
Part a) Show the binding naming the variables x,y,z (using R11 addressing)

```
x     EQU  -12 ;32-bit signed number
y     EQU  -8  ;32-bit signed number
z     EQU  -4  ;32-bit signed number
```

Part b) Show the allocation in assembly language

```
Func PUSH {R11,LR}
     MOV  R11,SP
     SUB  SP,#12     ;allocate x,y,z
```

Part c) Show assembly code that sets x=6

```
     MOV  R0,#6
     STR  R0,[R11,#x] ;x=6
```

Part d) Show assembly code that sets y=1.5*x (1.5 is 1 + ½)

```
     LDR  R0,[R11,#x] ;R0=x
     MOV  R1,R0
     ASR  R1,R1,#1   ;0.5x
     ADD  R0,R0,R1   ;1.5x
     STR  R0,[R11,#y]
```

Part e) Show the assembly code that deallocates the variables

```
     ADD  SP,#12     ;deallocation
```

```
POP  {R11,PC}
```

Answer 6. We need to store values from -100 to 100 cm with a resolution of at least 0.01cm. Choose **decimal fixed point** because there are a lot of human input/output and not much internal calculations. The fixed constant **is 0.01 (units cm).** The integers will range from -10000 to +10000, so choose **16-bit signed**.

Answer 7. We need to store values from -100 to 100 cm with a resolution of at least 0.01cm. Choose **binary fixed-point** because there are a lot of internal calculations and not much human input/output. The fixed constant is $\mathbf{2^{-7}}$ **(units cm),** because it is smaller than 0.01. The integers will range from -12800 to +12800, so choose **16-bit signed**.

Answer 8. Write a function in C, which uses fixed point calculations, to calculate the area of a circle. The input parameter is radius in mm, and the output is area in mm^2.
$A = \pi * r^2$. Approximate π by 3217/1024.

```
uint32_t Area(uint32_t r){
  return (3217 * r * r)>>10;
}
```

Answer 9a. List the meaning of these pins that connect the microcontroller to the LCD:

```
// pin 8  SCK          PA2 (SSI0Clk)  Serial clock
// pin 7  MOSI         PA5 (SSI0Tx)   Serial Data output to LCD
// pin 6  TFT_CS       PA3 (SSI0Fss)  LCD select (0 means enable)
// pin 4  D/C          PA6 (GPIO)     1=data, 0=command
// pin 3  RESET        PA7 (GPIO)     1=run, 0=reset
```

Answer 9b. List the meaning of these pins that connect the microcontroller to the SSD1306 OLED:

```
// SCL   PB2 I2C clock, the clock toggles for each bit sent
// SDA   PB3 I2C data, this line contains the data being sent
```

The clock is always sent by the master, which is the TM4C123
The data can go either direction. For the SSD1306, the data just goes from TM4C123 to SSD1306

Answer 10a. There are four steps to output data to the LCD. Explain the purpose of each step.

;1) Read SSI0_SR_R and check bit 1,
To send data to the LCD, we implement busy-wait synchronization. First we read the status bit. If the status is 0, the interface is busy. If the status is 1, the interface is ready
;2) If bit 1 is low loop back to step 1
Steps 1+2 are the essence of busy-wait synchronization. We loop (wait) until the interface is ready. Because the SSI interface is fast, this wait will be very short, less than 2us.
;3) Set D/C=PA6 to one
1 means data. This means the transmitted bits will be considered as data to the LCD.
;4) Write the 8-bit data to SSI0_DR_R
This will write data into the SSI and begin 8-bit transmission to the LCD.

Answer 10b. These are the first are four steps to output data to the SSD1306 OLED. Explain the purpose of each step.

```
; 1) wait while I2C is busy, wait for I2C3_MCS_R bit 0 to be 0
```

To send data to the OLED, we implement busy-wait synchronization. First we read the status bit. If the status is 1, the interface is busy. If the status is 0, the interface is ready

```
; 2) write slave address to I2C3_MSA_R,
;     MSA bits7-1 is slave address
;     MSA bit 0 is 0 for send data
```

Since there could be multiple slaves on the I2C bus, the protocol requires sending an address at the start of each communication. The 7-bit address is placed in bits 7-1. Since I2C supports reading from and writing to the peripheral, we must specify if this communication writes data from master to slave (bit0=0), or reads data from slave to master (bit0=1).

```
; 3) write first data to I2C3_MDR_R
```

This will write data into the I2C interface (but not yet send it).

```
; 4) write 0x03 to I2C3_MCS_R,  send no stop, generate start, enable
```

This will begin transmission to the OLED: address and first data.

~~**Answer 11a**. Abstraction means separation of what it does (see prototypes in **ST7735.h**) from how it works (see C functions in **ST7735.c** file).~~

Answer 11b. Abstraction means separation of what it does (see prototypes in **SSD1306.h**) from how it works (see C functions in **SSD1306.c** file).

Answer 12. This function will allocated an array on the stack.

Part a) Show the binding naming the variables x,y,z (using SP addressing)

```
buf   EQU  0  ;100 16-bit numbers
```

Part b) Show the allocation in assembly language

```
Func SUB  SP,#200     ;allocate 100 16-bit numbers
```

Part c) Show assembly code that sets buf[6]=6

```
     MOV  R0,#6
     LSL  R1,R0,#2   ;offset is 12
     ADD  R2,R1,#buf ;base+2*index
     STRH R0,[SP,R2] ;buf[i]=i
```

Part d) Show assembly code that sets buf[i]=i

```
     MOV  R0,#0  ;i
Loop LSL  R1,R0,#2
     ADD  R2,R1,#buf ;base+2*index
     STRH R0,[SP,R2] ;buf[i]=i
     ADD  R0,R0,#1
     CMP  R0,#100
     BLO  Loop
```

Part e) Show the assembly code that deallocates the variables

```
     ADD  SP,#200    ;deallocation
     BX   LR
```

Answer 13. The assembly pseudo-op **IMPORT** allows this assembly code to access variables or functions in other file. Those other variables or functions may be other assembly or C.

Answer 14. The assembly pseudo-op **EXPORT** allows other software to access variables or functions in this file. The other software may be assembly or C.

Answer 15. Value = integer*resolution. The range of 8-bit unsigned integers is 0 to 255. The resolution is 10^-3, which is 1/1000, or 0.001. Therefore the range of values is 0 to 0.255.

Answer 16. Value = integer*resolution. The range of 8-bit signed integers is -128 to +127. The resolution is 2^-3, which is 1/8, or 0.125. Therefore the range of values is -128/8 to 127/8, or -16 to +15.875.