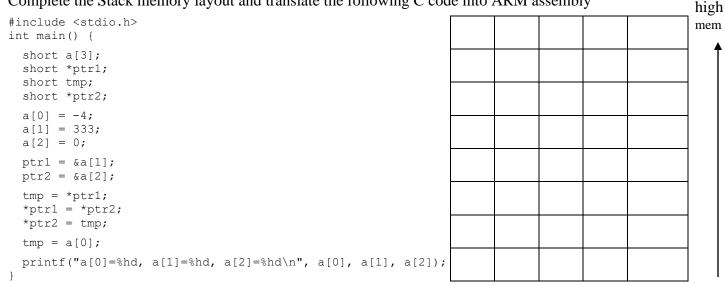
## **CSE 30**

## **ARM Architecture Memory / Data Type Alignments**

Complete the Stack memory layout and translate the following C code into ARM assembly



```
cortex-a53
 .cpu
 .syntax unified
 .section .rodata
fmt:
 .asciz "a[0]=%hd, a[1]=%hd, a[2]=%hd\n"
 .global main
 .text
 .align 2
main:
 .equ FP_OFF, 4
 .equ A, _____
 .equ
     PTR1, _____
     TMP, _____
 .equ
      PTR2, _____
 .equ
     PAD, ___
 .equ
     FRMADD, _____
 .equ
 push
       {fp, lr}
 add
      fp, sp, ____
   @ a[0] = -4, a[1] = 333, a[2] = 0
      r3,
 _____ r3, [fp, ____] @ a[0]
 _____ r3, _
  _____ r3, [fp, _____] @ a[1]
      r3, ____
 mov
 _____ r3, [fp, ____] @ a[2]
```

```
@ ptr1 = &a[1]
     r3, ____, __
   ___ r3, [fp, ____]
@ ptr2 = &a[2]
    r3, _____, ____
add
_____ r3, [fp, ____]
@ tmp = *ptr1
ldr
    r3, [ _____]
_____ r3, [____]
strh r3, [____]
@ *ptr1 = *ptr2
    r3, [____]
   ___ r3, [____]
ldr
    r2, [____
____ r3, [ ]
@ *ptr2 = tmp
ldrsh r3, [____]
                      @ tmp
      r2, [____]
ldr
strh
      _____, [____]
0 \text{ tmp} = a[0]
ldrsh r3, [_____
strh r3, [fp, ____]
@ print values
ldrsh ____, [____
```

```
ldrsh _____, [____] bx lr
ldrsh _____, [____] .end
bl printf
sub sp, fp, FP_OFFSET
pop {fp, lr}
```

\_\_\_\_\_\_

## Translate the following C code into Assembly. Assume all variables and parameters are stored on the stack.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
int a = 20;
unsigned short b = 10;
short c;
int dosth(int a0, int a1, int a2, int a3, short a4, unsigned short a5) {
    return a0 + a1 + a2 + a3 + a4 + a5;
}
int
main(int argc, char* argv[]){
    int arr[SIZE];
    FILE *f = fopen(argv[1], "r");
    if (f==NULL) {
       perror("error open file");
        return -1;
    }
    for(int i = 0; i < SIZE; ++i){</pre>
        fscanf(f, "%d", &arr[i]);
        printf("%d", arr[i]);
    }
    a = dosth(arr[0], arr[1], arr[2], arr[3], c, b);
    printf("%d", a);
    return 0;
}
                                                  //create c
      .cpu cortex-a53
      .syntax unified
      .arch armv6
      .extern fopen
    .extern perror
    .extern fscanf
    .extern printf
//create a and b
                                                         .section .rodata
                                                               .string "r"
                                                  .Lfmt1:
                                                  .Lfmt2:
                                                               .string "error open file"
                                                  .Lfmt3:
                                                               .string "%d"
```

```
.section .text
                                          //start for loop
     .align 2
     .global main
     .type main, %function
     .equ SIZE, _____
     .equ FP_OFF, _____ //need r4
     .equ ARR, _____
     .equ F, _____
     .equ I, _____
     .equ PAD, _____
                                          //use r4 as a backup for &arr[i]
     .equ OARG6, _____
                                          add r4, fp, -ARR
     .equ OARG5, _____
                                      .Lfor:
     .equ FRMADD, _____
main: #create stack frame
    push {_____ fp, lr}
    add fp, sp, FP OFF
    # FILE *f = fopen(argv[1], "r");
    //if (f == NULL){.....}
                                      .Lendfor:
                                     //call a = dosth(arr[0], arr[1], arr[2],
                                      arr[3], c, b)
```

.Lendif:

```
//call printf("%d", a);
      mov r1, r0
      ldr r0, =.Lfmt3
      bl printf
      b .LreturnS
.LreturnF:
      mov r0, -1
      b .Ldone
.LreturnS:
      mov r0, 0
.Ldone:
      sub sp, fp, FP_OFF
      pop {r4, r5, fp, lr}
      #go back to the caller
      bx lr
.size main, (. - main)
      .section .text
      .align 2
      .global dosth
      .type dosth, %function
     .equ FP_OFF, ______
//no preserved is used
      .equ ARG5, _____
      .equ ARG6, _____
```

dosth:

```
//return
sub sp, fp, FP_OFF
pop {fp, lr}
bx lr

.size dosth, (. - dosth)
.section .note.GNU-stack, "", %progbits
.end
```