**CSE 30**

**ARM Architecture Memory / Data Type Alignments**

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

high

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  |  | |  | mem |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  | |  | |  |  | |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |
|  |  | |  | | |  |  |  |

#include <stdio.h>

int main() {

short a[3];

short \*ptr1;

short tmp;

short \*ptr2;

a[0] = -4;

a[1] = 333;

a[2] = 0;

ptr1 = &a[1];

ptr2 = &a[2];

tmp = \*ptr1;

\*ptr1 = \*ptr2;

\*ptr2 = tmp;

tmp = a[0];

printf("a[0]=%hd, a[1]=%hd, a[2]=%hd\n", a[0], a[1], a[2]);

}

.cpu cortex-a53

.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, \_\_\_\_\_\_\_\_\_\_\_\_\_

.equ TMP, \_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ PTR2, \_\_\_\_\_\_\_\_\_\_\_\_\_

.equ PAD, \_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ FRMADD, \_\_\_\_\_\_\_\_\_\_\_

push {fp, lr}

add fp, sp, \_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_** \_\_\_\_\_\_, \_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_

**@ a[0] = -4, a[1] = 333, a[2] = 0**

mov r3, **\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_** r3, [fp, \_\_\_\_\_\_\_\_] @ a[0]

\_\_\_\_\_\_\_ r3, \_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_** r3, [fp, **\_\_\_\_\_\_\_\_**] @ a[1]

mov r3, \_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_** r3, [fp, **\_\_\_\_\_\_\_\_**] @ a[2]

**@ ptr1 = &a[1]**

add r3, **\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_** r3, [fp, **\_\_\_\_\_\_\_\_**]

**@ ptr2 = &a[2]**

add r3, **\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_** r3, [fp, **\_\_\_\_\_\_\_\_**]

**@ tmp = \*ptr1**

ldr r3, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**]

**\_\_\_\_\_\_\_** r3, [\_\_\_\_\_\_\_]

strh r3, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**]

**@ \*ptr1 = \*ptr2**

ldr r3, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**] @ ptr2

**\_\_\_\_\_\_\_** r3, [\_\_\_\_\_\_\_]

ldr r2, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**] @ ptr1

**\_\_\_\_\_\_\_** r3, [\_\_\_\_\_\_\_]

**@ \*ptr2 = tmp**

ldrsh r3, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**] @ tmp

ldr r2, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**] @ ptr2

strh \_\_\_\_\_\_\_, [\_\_\_\_\_\_\_]

**@ tmp = a[0]**

ldrsh r3, [**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**]

strh r3, [fp, **\_\_\_\_\_\_\_\_**]

**@ print values**

\_\_\_\_\_\_ \_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_

ldrsh \_\_\_\_\_\_\_, [\_\_\_\_\_\_\_\_\_\_\_\_\_\_]

ldrsh \_\_\_\_\_\_\_, [\_\_\_\_\_\_\_\_\_\_\_\_\_\_]

ldrsh \_\_\_\_\_\_\_, [\_\_\_\_\_\_\_\_\_\_\_\_\_\_]

bl printf

sub sp, fp, FP\_OFFSET

pop {fp, lr}

bx lr

.end

====================================================================================

**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){

        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;

}

.cpu cortex-a53

.syntax unified

.arch armv6

.extern fopen

.extern perror

.extern fscanf

.extern printf

//create a and b

//create c

.section .rodata

.Lfmt1: .string "r"

.Lfmt2: .string "error open file"

.Lfmt3: .string "%d"

.section .text

.align 2

.global main

.type main, %function

.equ SIZE, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ FP\_OFF, \_\_\_\_\_\_\_\_\_\_\_\_\_\_ //need r4

.equ ARR, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ F, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ I, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ PAD, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ OARG6, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ OARG5, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.equ FRMADD, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

main: #create stack frame

push {\_\_\_\_\_\_\_\_\_\_\_\_\_ fp, lr}

add fp, sp, FP\_OFF

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

# FILE \*f = fopen(argv[1], "r");

//if (f == NULL){……}

.Lendif:

//start for loop

//use r4 as a backup for &arr[i]

add r4, fp, -ARR

.Lfor:

.Lendfor:

//call a = dosth(arr[0], arr[1], arr[2], arr[3], c, b)

//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