EED 3018 Microprocessor Systems

Term Homework

Objective: To convert the given C++ code into MSP430 assembly.

Due Date: 29.5.2020

A converter program will be developed to read and convert a C program into a MSP430 assembly software. The converter program will be an executable file, you can use any language to develop your software, it will read a file with an extension *.cpp or *.c (the file name is entered by the user, it is in the same directory with the software) and convert it into a file with an extension *.asm. The asm file is the MSP430 assembly correspondent of the C file meaning that its functionality will be the same.

C program can include the following functions:

Arithmetic and logical functions:

+,-,*,*AND,OR,XOR,=,<,>*

Control functions:

If, else, else if, for

Example:

The sample program is:

```
int fun(int a, int b)
{
    return a + b;
}

main()
{
    int m;

    m = fun(1, 2);
}
```

The procedure is:

1. The current value of r4 (used as **frame pointer** in MSP430 family) is pushed into the stack. The **stack pointer** (r1) automatically gets decremented by 2.
2. r1 gets decremented again by an offset, thus allocating a stack frame. The offset by which r1 gets decremented depends on the number of local variables in the called function.

   Here, r1 gets decremented by 4, since there are two local variables for *fun()*, *a* and *b*.
3. The current value of the stack pointer r1 is copied into r4 (the register r4 thus indicates the frame pointer for the currently executing function).

All further manipulations of the local variables will be with reference to the frame pointer r4.

4. After the body of the called function is executed, the same offset as above is added back to the stack pointer r1, thus deallocating the stack frame.

5. The current value of r1 is popped into r4, thus retrieving the previous stack frame. The stack pointer r1 gets auto-incremented by 2.

```
fun:
    push    r4
    sub #4, r1
    mov r1, r4
    /* prologue ends here (frame size = 4) */
.L__FrameSize_fun=0x4
.L__FrameOffset_fun=0x6
    mov r15, @r4
    mov r14, 2(r4)
    mov @r4, r15
    add 2(r4), r15

    /* epilogue: frame size = 4 */
    add #4, r1
    pop r4
    ret
```

The assembler directives __**FrameSize** and __**FrameOffset** gives the size and offset of the frame allocated for the function *fun()*.

Other examples can be found at http://harijohnkuriakose.blogspot.com/2010/11/translating-c-constructs-to-msp430.html

Each software will be uploaded to Sakai online education system and will be graded.

Good Luck