Lab 5 The C and V Flags of an ARM MCU

Lab Report

Student: Jeremiah Webb

Student ID: 2545328

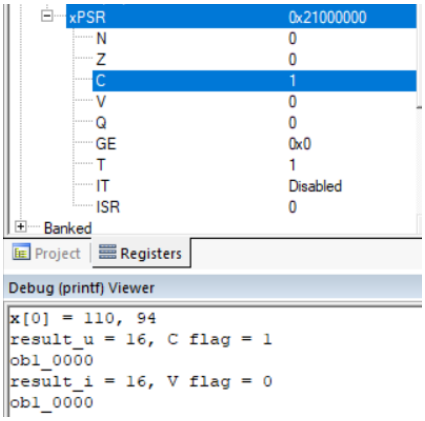
Instructor: Dr. Jianhua Liu

Section #2

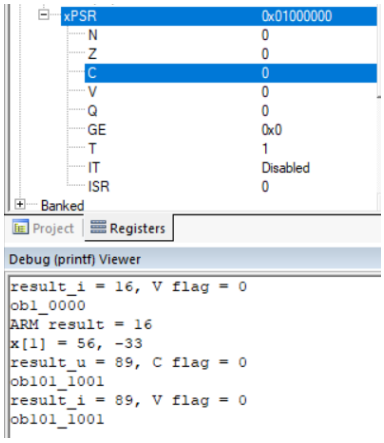
Introduction

Understanding not only the arithmetic expression and use of the C and V flags but also understanding how C and V flags are used in programming is crucial to the full comprehension of the flags. Being able to not only understand it on paper through math, but logically will help students apply C and V flags throughout the class.

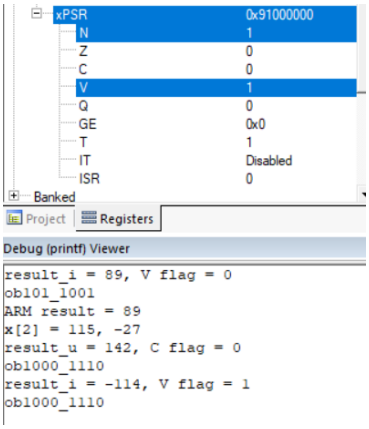
Screenshots



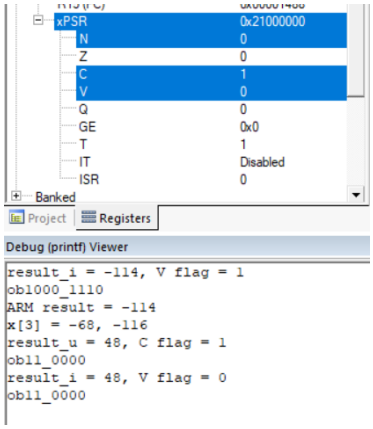
Results for x[0]



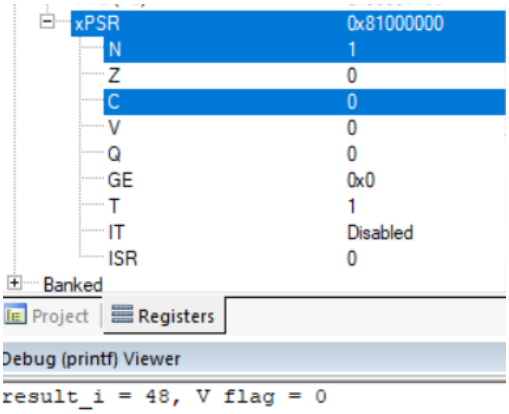
Results for x[1]



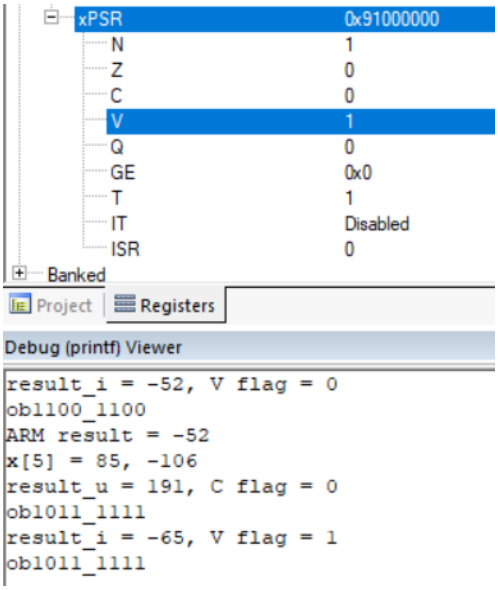
Results for x[2]



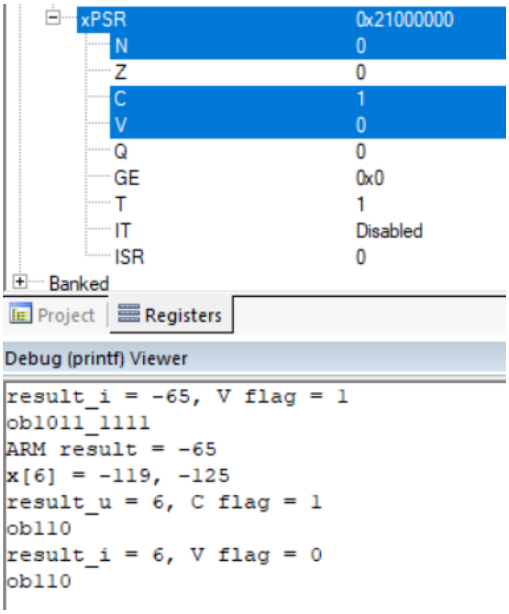
Results for x[3]



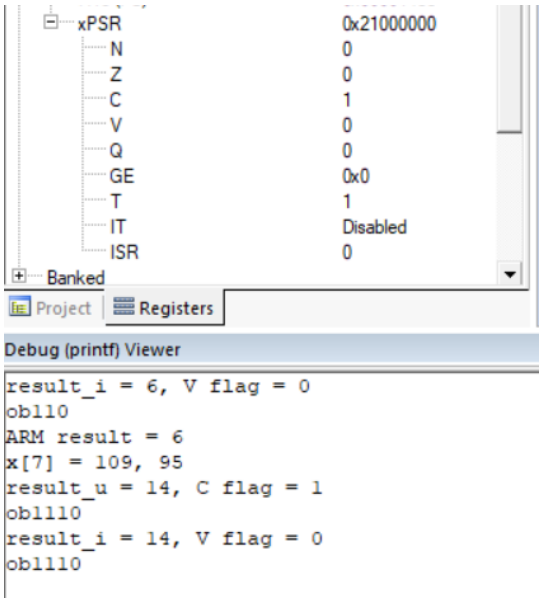
Results for x[4]



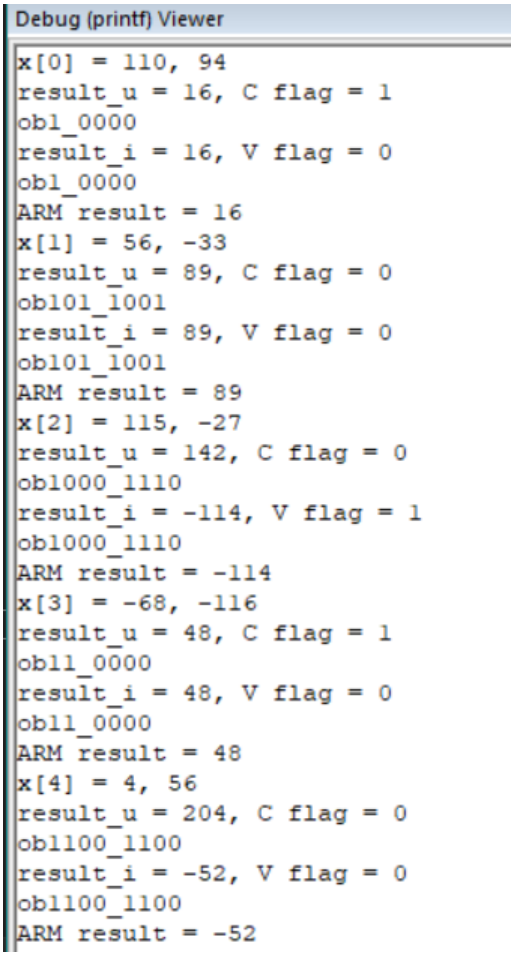
Results for x[5]



Results for x[6]



Results for x[7]



Debug (printf) view of results. C and V flags included.

Code Snippets

```
// return x0 - x1
uint8_t sub_uint8(uint8_t x0, uint8_t x1, bool *c_flg){
     if(x0 > x1)
     {
          *c_flg = 1;
     }
     else{
          *c_flg = 0;
     }
return((x0 - x1));
}

// return x0 - x1
int8_t sub_int8(int8_t x0, int8_t x1, bool *v_flg){
//Range [-2^n-1, 2^n-1 -1] [-128, 127]
     int result = x0 - x1;
     int limitbot = -128;
     int limittop = 127;

     if(result < limitbot || result > limittop){
          *v_flg = 1;
     }
     else{
          *v_flg = 0;
     }
     return((x0 - x1));
}
```

Questions

Explain the specific operations to generate `x[i][0]` and `x[i][1]`.

To develop numbers x[i][0] and x[i][1] they are both equal to the rand () function, which generates a pseudo random number, then does a modulus operator with MAX_Un, which is 8 shifted to the left 1, and subtracted by 1. Then that remainder is added by MIN_IN, which is 8 subtracted by 1, shifted to the left by 1, and assigned a negative sign. After doing these operations to both x[i][0] and x[i][1], they are used in functions sub_uint8 and sub_int8.

Narrative

Overall, the lab went well, the biggest issue I had was creating the V flag function to determine which operations need the V flag to be turned on. However, after researching and seeing that using specific limits to see if a math function needs overflow, the logic was then easy to implement within the sub_int8 function. This lab did assist with my understanding of how to implement such logic into C.

Results

As shown in the screenshots, the C and V flags are properly printed and represented and match the C and V flags in the xPSR. Additionally, the code given works properly and presents both signed and unsigned results of the numbers generated. Binary is also printed out properly in the verbose form.