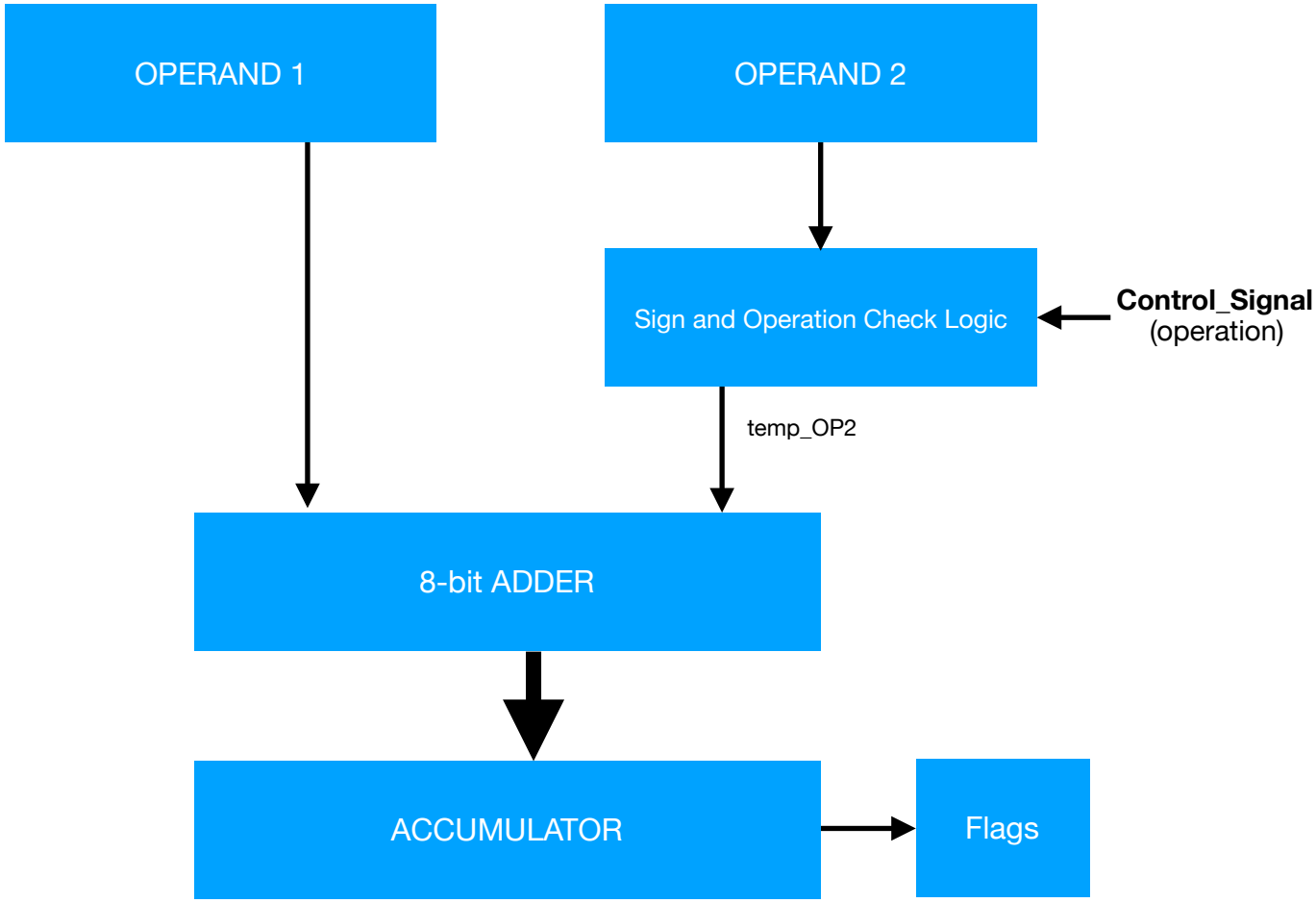
ADDER/SUBTRACTOR



Note:

- 2's complement operand2 only if the operation is ADD.
- Accumulator (ACC) is 8-bits wide. But it has to be declared as **unsigned int** in order to detect carry conditions.

Solution:

- Create a variable of type **unsigned char** called temp_OP2 (this would be the 2's complemented value for operand2).
- Global variables SF (sign flag), OF (overflow flag), (zero flag) ZF, (carry flag) CF. The accumulator (ACC) should be local. (Warning: do not initialize ACC with any value.)
- Write a function called setFlags(int ACC) to set the flags (SF, ZF, OF, CF).
- Below is a snippet inside your ALU( ) function:

```
/* this function represents the ALU */
int ALU(unsigned char operand1, unsigned char operand2, unsigned char control_signal)
{
  static unsigned int ACC;
  unsigned char temp_OP1=0x00, temp_OP2=0x00;

  /* setting ACC and flags to initial values */
  ACC = 0x0000; SF=0, CF=0, ZF=0 OF=0;
  …
  …
  …

  if(control_signal==0x01 || control_signal==0x02)   // ADD or SUB
  {
     /* Sign and Operation Check Logic */
     temp_OP1 = operand1;

     if(control_signal==0x02)                        // check if operation is SUB
         temp_OP2 = twosComp(operand2);              // 2's complement operand2
     else
         temp_OP2 = operand2;

     /* 8-bit Adder */
     ACC = temp_OP1 + temp_OP2;
  }
  else if(control_signal==0x02)                     // MUL
  {
     …
     …
     …
  }
  setFlags(ACC);                                    // set the flags
}
```

Solution for setFlags( ):

- Function prototype for setFlags: void setFlags(unsigned int ACC).
- ZF = 1 if the result of the operation is 0 otherwise 0.
- SF = 1 if the sign bit of the ACC is 1 (negative) otherwise 0. The sign bit is the 8th bit of ACC (not the 16th).
- OF = 1 if ACC is greater than 0x7F (127 or 2^{8-1}) which is the last positive number. Any number higher is already negative.
- CF = 1 if ACC is greater than 0xFF (255 or 2^{8}). Any number is outside the range -128 to +127. This flag will also be set as “carry bit” for the shift operations (SHL & SHR)

```
/* this function sets the flags after the arithmetic or logical operation */
void setFlags(unsigned int ACC)
{
  if(ACC==0x0000)  // check if value of ACC is equal to 0
     ZF = 1;

  if((ACC & 0x0080)==0x0080) // check if sign (8th bit) of ACC is 1 (negative)
     ...
     ...
     ...
}
```

More hints:

- To display the operands and ACC as binary, write a function called printBin(unsigned char data, unsigned char data_width). It will print the bits as characters.
- Declare ACC as *unsigned int* (16-bit), global variable. Always use casting for assigning 8-bit values to a 16-bit variable.