# Some C basics

- Every C program has a function main()
  - The simplest C program is:

```
main()
{
}
```

- Every statement ends with a semicolon

```
x = a+b;
```

- Comment starts with `/*` ends with `*/`

```
/* This is a comment */
```

- Simple program – increment Port A

```
#include "hc12b32.h"

main()
{
    DDRA  = 0xff;  /* Make PORTA output */
    PORTA = 0;     /* Start at 0 */
    while(1)       /* Repeat forever */
    {
        PORTA = PORTA + 1;
    }
}
```

- Data Types:

| 8-bit | 16-bit |
|---|---|
| unsigned char | unsigned int |
| signed char | signed int |

- Need to declare variable before using it:

```
signed char c;
unsigned int i;
```

- Can initialize variable when you define it:

```
signed char c = 0xaa;
signed int i = 1000;
```

  - You tell compiler it you are using signed or unsiged numbers; the compiler will figure out whether to use BGT or BHI

- Arrays:

```
unsigned char table[10];  /* Set aside 10 bytes for table */
```

  - Can refer to elements `table[0]` through `table[9]`
  - Can initialize an array when you define it:

```
table[] = {0xaa, 0x55, 0xa5, 0x5a};
```

- Arithmetic operators:

```
+  (add)           x = a+b;
-  (subtract)      x = a-b;
*  (multiply)      x = a*b;
/  (divide)        x = a/b;
%  (modulo)        x = a%b;  (Remainder on divide)
```

- Logical operators

```
&  (bitwise AND)                y = x & 0xaa;
|  (bitwise OR)                 y = x | 0xaa;
^  (bitwise XOR)                y = x ^ 0xaa;
<< (shift left)                 y = x << 1;
>> (shift right)                y = x >> 2;
~  (1's complement)             y = ~x;
-  (2's complement - negate)    y = -x;
```

Check for equality - use `==`

```
if (x == 5)
```

Check if two conditions true:

```
if ((x==5) && (y==10))
```

Check if either of two conditions true:

```
if ((x==5) || (y==10))
```

- Assign a name to a number

```
#define COUNT 5
```

- Include a header file (such as `hc12b32.h`:

```
#include "hc12b32.h"
```

- Declare a function: Tell what parameters it uses, what type of number it returns:

```
int read_port(int port);
```

- If a function doesn't return a number, declare it to be type void

```
void delay(int num);
```

## Setting and Clearing Bits using Assembly and C

- To put a specific number into a memory location or register (e.g., to put 0x55 into PORTA):
  - In assembly:

```
ldaa    #$55
staa    PORTA
```

  - In C:

```
PORTA = 0x55;
```

- To set a particular bit of a register (e.g., set Bit 4 of PORTA) while leaving the other bits unchanged:
  - In assembly, use the `bset` instruction with a mask which has 1's in the bits you want to set:

```
bset    PORTA,#$10
```

  - In C, do a bitwise OR of the register with a mask which has 1's in the bits you want to set:

```
PORTA = PORTA | 0x10;
```

- To clear a particular bit of a register (e.g., clear Bits 0 and 5 of PORTA) while leaving the other bits unchanged:
  - In assembly, use the `blcr` instruction with a mask that has 1's in the bits you want to clear:

```
bclr    PORTA,#$21
```

  - In C, do a bitwise AND of the register with a mask that has 0's in the bits you want to clear:

```
PORTA = PORTA & 0xDE;
```

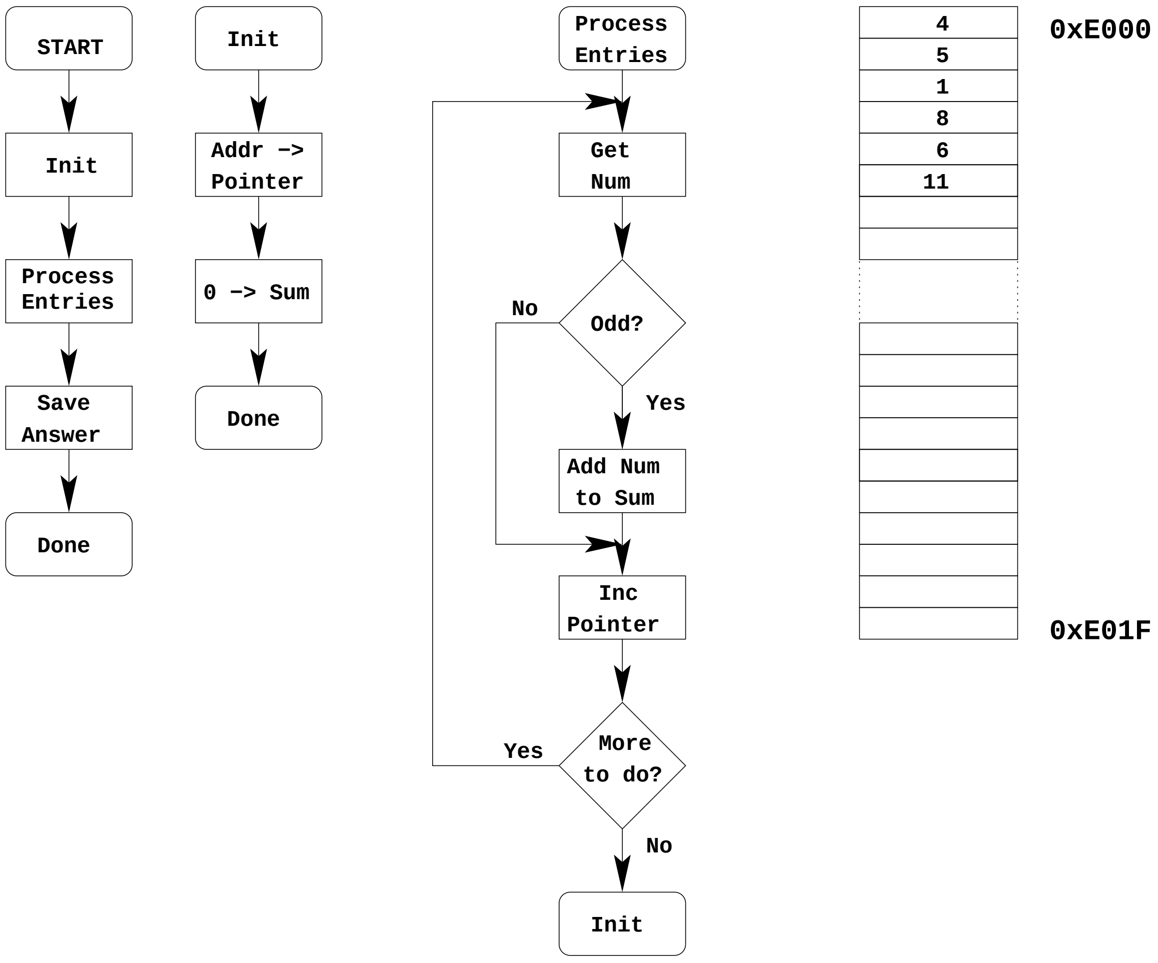
or

```
PORTA = PORTA & ~0x21;
```

## Sum the odd 8-bit numbers in an array

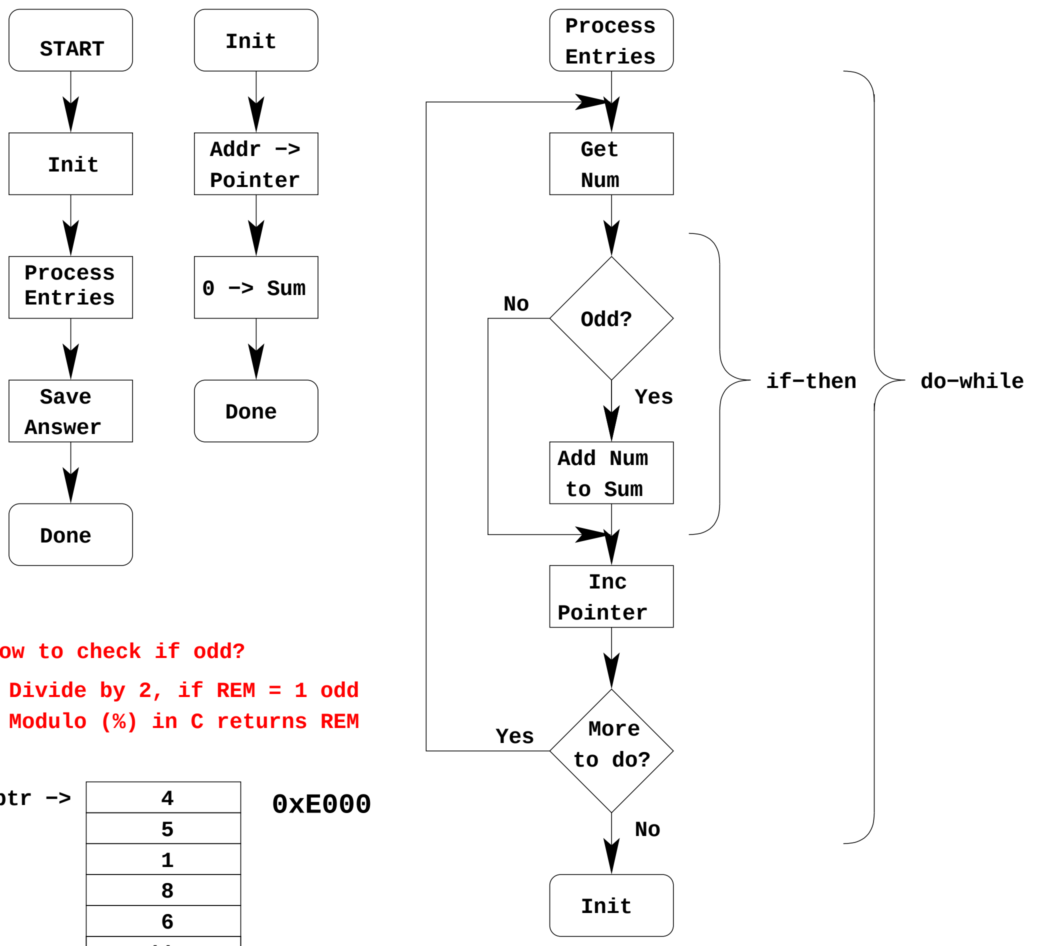
- Write a program to sum all the odd numbers in an array of data.
- The numbers in the array should be treated as unsigned 8-bit numbers.
- The array starts at address `0xE000` and ends at address `0xE01F`.
- This is the same program which was done in assembly language on Feb. 1, 2002.

SUM ODD 8-BIT NUMBERS IN ARRAY FROM 0xE000 TO 0xE01f



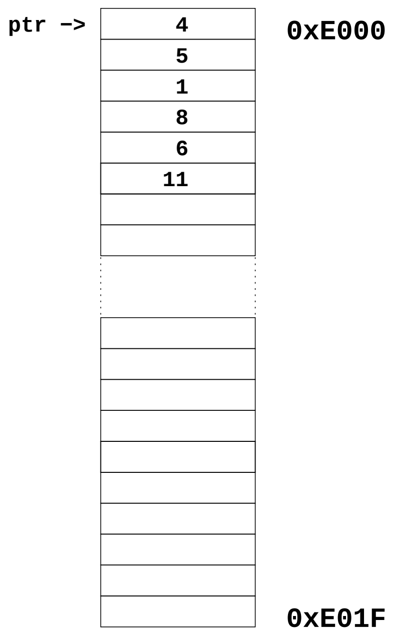
## Convert to C

SUM ODD 8-BIT NUMBERS IN ARRAY FROM 0xE000 TO 0xE01f



How to check if odd?

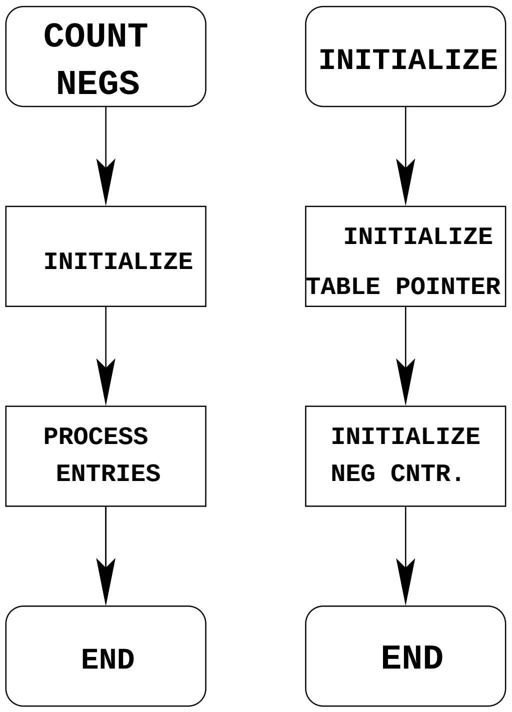
Divide by 2, if REM = 1 odd
Modulo (%) in C returns REM

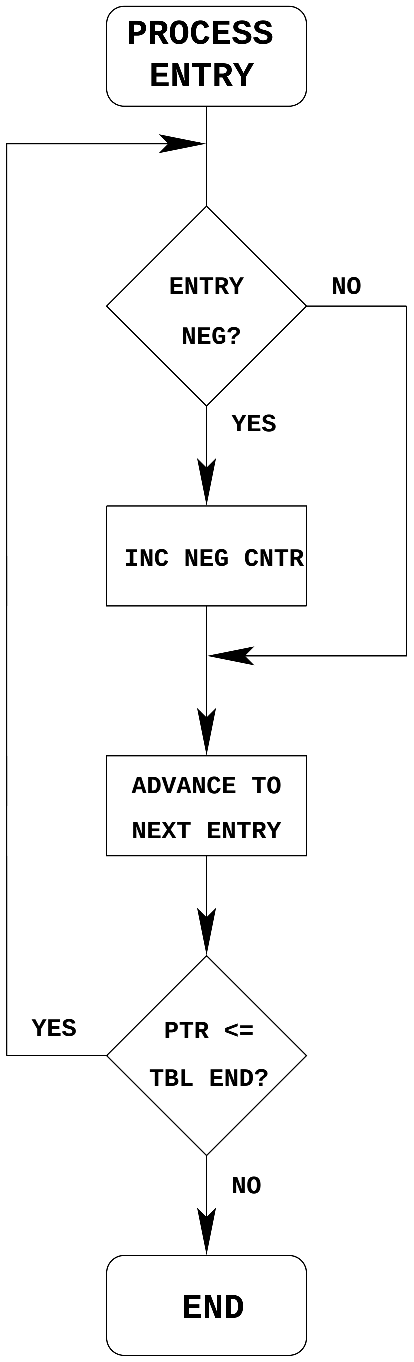


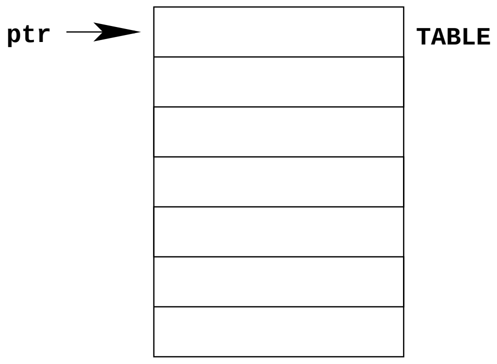
```
main() {
   ptr = (unsigned char *) 0xe000;
   sum = 0;
   do {
      x = *ptr;
      if ((x % 2) == 1) {
          sum = sum + x;
      }
      ptr = ptr + 1;
   }
   while (ptr <= (unsigned char *) 0xe01f);
}
```

```
/* Program to sum the odd numbers in an array
 * The numbers are unsigned characters
 * The array starts at address 0xE000 and
 * ends at adress 0xE01F
 */

#define START 0xE000
#define END   0xE01F

main()
{
    unsigned int sum;    /* Keep the running sum */
    unsigned char *ptr;  /* Pointer to array element */
    unsigned char x;     /* Character from array */

    ptr = (unsigned char *) START;
    sum = 0;
    do
    {
        x = *ptr;            /* Get entry */
        if ((x % 2) == 1)    /* Is number odd? */
        {
            sum = sum + x;   /* Odd:  add to sum */
        }
        ptr = ptr + 1;       /* Advance to next */
    }
    while (ptr <= (unsigned char *) END);  /* Done? */
}
```

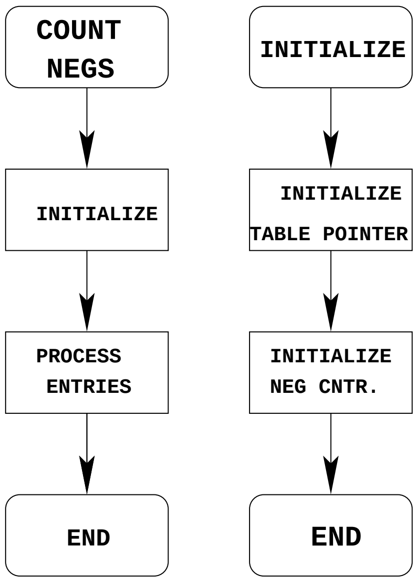
## Count the number of negative numbers in an array



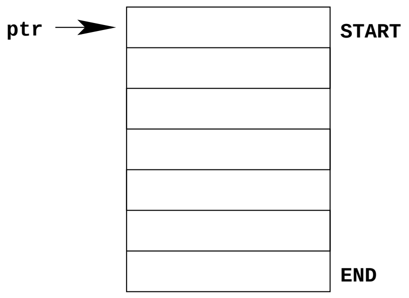


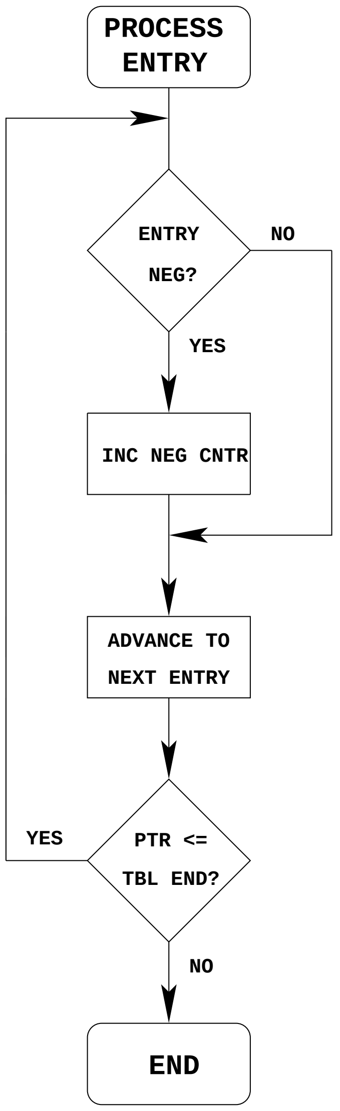


## Convert to C



```
ptr = START;

count = 0;
```





```
do {

   if (*ptr < 0) {

         count = count + 1;

   }

    ptr = ptr + 1;

}

while (ptr <= END);
```

```
/*
 * Program to count the number of
 * 8-bit negative numbers from
 * 0x8000 to 0xBFFF
 */

#define START 0x8000
#define END   0xBFFF

main()
{
    int count;
    signed char *ptr;

    count = 0;
    ptr = (signed char *) START;
    do
    {
        if (*ptr < 0)
        {
            count = count + 1;
        }
        ptr = ptr + 1;
    }
    while (ptr <= (signed char *) END);
}
```

## A software delay

- To enter a software delay, put in a nested loop, just like in assembly.
  - Write a function `delay(num)` which will delay for `num` milliseconds

```
void delay(unsigned int num)
{
    unsigned int i;

    while (num > 0)
    {
        i = XXXX;
                       /* ----------------------- */
        while (i > 0)  /*                         */
        {              /* Want inner loop to delay */
            i = i - 1; /* for 1 ms                */
        }              /*                         */
                       /* ----------------------- */
        num = num - 1;
    }
}
```

- What should `XXXX` be to make a 1 ms delay?

- Look at assembly listing generated by compiler:

```
          ; 27 void delay(int num)
          ; 28 {
              switch    .text
          _delay:
              pshd
              pshd
          OFST:    set    2
              bra    L55
          L35:
          ; 33          i = XXXX;
             ldy    #XXXX
       ------------------------------------------------------------
inner  | L16:
loop   | ; 36                i = i-1;
takes  |     dey
6      |     sty    OFST-2,s
cycles | ; 34          while (i > 0)
       |     bgt    L16
       ------------------------------------------------------------
         ; 38          num = num - 1;
             ldy    OFST+0,s
             dey
             sty    OFST+0,s
         L55:
         ; 31     while (num>0)
             ldd    OFST+0,s
             bgt    L35
         ; 40 }
             leas    4,s
             rts
             xdef    _main
             xdef    _delay
```

- Inner loop takes 6 cyles.
- One millisecond takes 8,000 cycles
  (8,000,000 cycles/sec $\times$ 1 millisecond = 8,000 cycles)
- Need to execute inner loop 8,000/6 = 1,333 times to dealy for 1 millisecond

```
void delay(unsigned int num)
{
    unsigned int i;

    while (num > 0)
    {
        i = 1333;
                       /* ------------------------  */
        while (i > 0)  /*                           */
        {              /* Inner loop takes 6 cycles */
            i = i - 1; /* Execute 1333 times to     */
        }              /* delay for 1 ms            */
                       /* ------------------------  */
        num = num - 1;
    }
}
```

## Program to increment LEDs connected to PORTA, and delay for 50 ms between changes

```
#include "hc12b32.c"

void delay(unsigned int num);
main()
{
    DDRA = 0xff;      /* Make PORTA output */
    PORTA = 0;        /* Start with all off */
    while(1)
    {
        PORTA = PORTA + 1;
        delay(50);
    }
}

void delay(unsigned int num)
{
    unsigned int i;

    while (num > 0)
    {
        i = 1333;
        while (i > 0)
        {
            i = i - 1;
        }
        num = num - 1;
    }
}
```

## Program to display a particular pattern of lights on PORTA

```
#include "hc12b32.c"

#define TABLEN 8

void delay(unsigned int num);
main()
{
    const char table[] = {0x80,0x40,0x20,0x10,
                          0x08,0x04,0x02,0x01};
    int i;

    DDRA = 0xff;      /* Make PORTA output */
    PORTA = 0;        /* Start with all off */
    i = 0;
    while(1)
    {
        PORTA = table[i];
        delay(50);
        i = i + 1;
        if (i >= TABLEN) i = 0;  /* Start over when */
                                 /* end is reached  */
    }
}
```