

- **Another simple program in assembly language**
- **Using the stack and the stack pointer**
- Huang Section 4.3
  - A program to add all the odd numbers in a memory array
  - Flow charts
  - Assembly language program
  - Assembly listing file
  - Assembly map file
  - The Stack and the Stack Pointer
  - The stack is an area of memory used for temporary storage
  - The stack pointer points to the last byte pushed onto the stack
  - Some instructions which use the stack, and how data is pushed onto and pulled off of the stack.

### Input and Output Ports

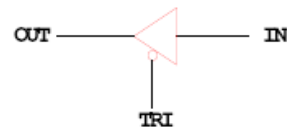
- How do you get data into a computer from the outside?

Any read from address \$0000 gets signals from outside

`LDAA $00`

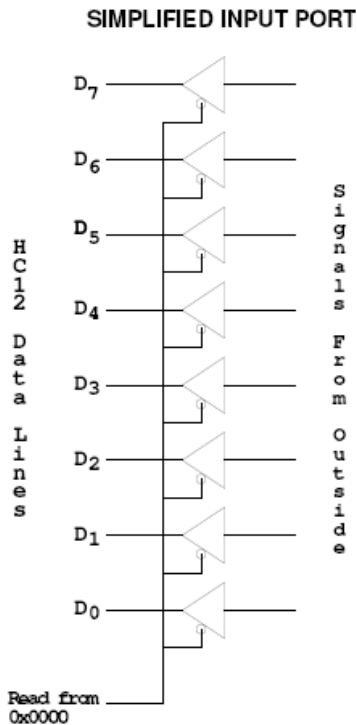
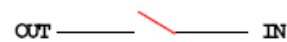
Puts data from outside into accumulator A.

Data from outside looks like a memory location.

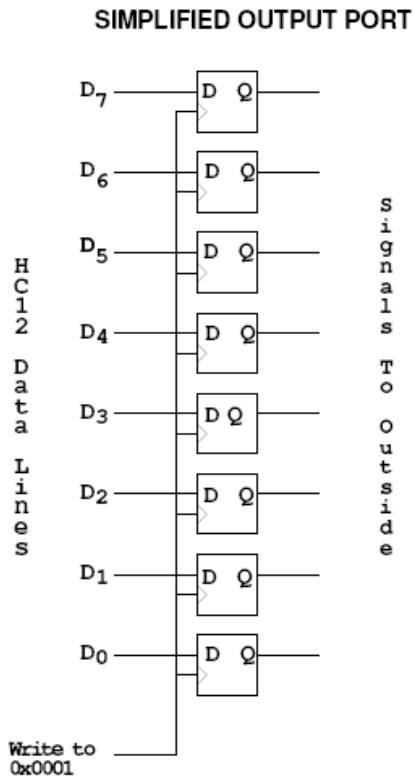


A Tri-State Buffer acts like a switch

If TRI is not active, the switch is open:  
OUT will not be driven by IN  
Some other device can drive OUT



- How do you get data out of computer to the outside?



Any write to address \$01 latches data into FF, so data goes to external pints

```
MOVB #$AA,$01
```

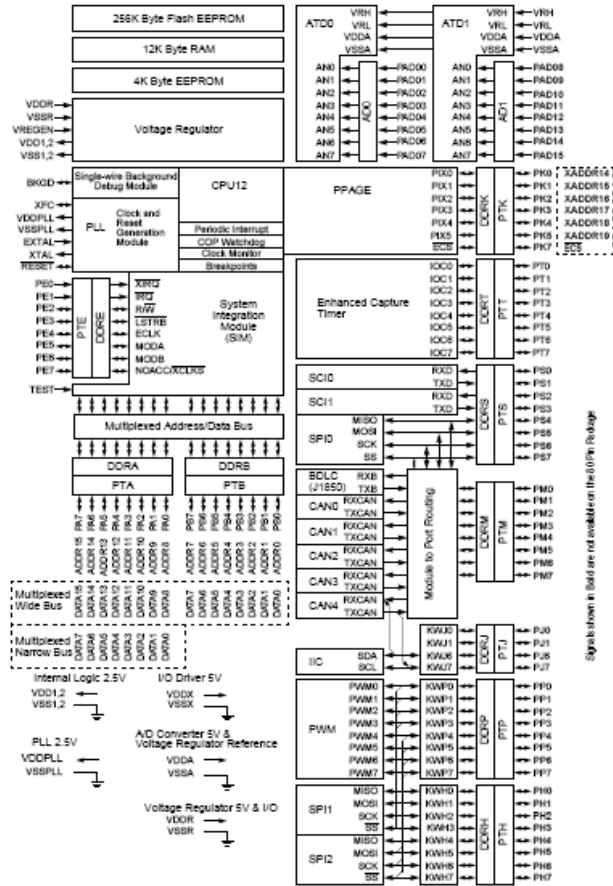
Puts \$AA on the external pins

When a port is configured as output and you read from that port, the data you read is the data which was written to that port:

```
MOVB #$AA, $01
LDAA $01
```

Accumulator A will have \$AA after this

Figure 1-1 MC9S12DP256B Block Diagram





For example, to make bits 3–0 of Port A input, and bits 7–4 output, write a 0xf0 to DDRA.

To send data to the output pins, write to PORTA (address 0x0000). When you read from PORTA input pins will return the value of the signals on them (0 => 0V, 1 => 5V); output pins will return the value written to them.

	<b>PA7</b>	<b>PA6</b>	<b>PA5</b>	<b>PA4</b>	<b>PA3</b>	<b>PA2</b>	<b>PA1</b>	<b>PA0</b>	
<b>Reset</b>	-	-	-	-	-	-	-	-	<b>\$0000</b>

Port B works the same, except DDRB is at address 0x0003 and PORTB is at address 0x0001.

```
;A simple program to make PORTA output and PORTB input,
;then read the signals on PORTB and write these values
;out to PORTA
```

```
prog:      equ   $1000
PORTA:    equ   $00
PORTB:    equ   $01
DDRA:     equ   $02
DDRB:     equ   $03

          org   prog
          movb  #$ff,DDRA   ; Make PORTA output
          movb  #$00,DDRB   ; Make PORTB input
          ldaa  PORTB
          staa  PORTA
          swi
```

• Because DDRA and DDRB are in consecutive address locations, you could make PORTA and output and PORTB and input in one instruction:

```
movw  #$ff00,DDRA      ; FF -> DDRA, 00 -> DDRB
```

### GOOD PROGRAMMING STYLE

1. Make programs easy to read and understand.
  - Use comments
  - Do not use tricks
2. Make programs easy to modify
  - Top-down design
  - Structured programming – no spaghetti code
  - Self contained subroutines
3. Keep programs short BUT do not sacrifice items 1 and 2 to do so

## TIPS FOR WRITING PROGRAMS

1. Think about how data will be stored in memory.
  - Draw a picture
2. Think about how to process data
  - Draw a flowchart
3. Start with big picture. Break into smaller parts until reduced to individual instructions
  - Top-down design
4. Use names instead of numbers

### Another Example of an Assembly Language Program

- Add the odd numbers in an array of data.
- The numbers are 8-bit unsigned numbers.
- The address of the first number is \$E000 and the address of the final number is \$E01F.
- Save the result in a variable called answer at address \$2000.

Start by drawing a picture of the data structure in memory:

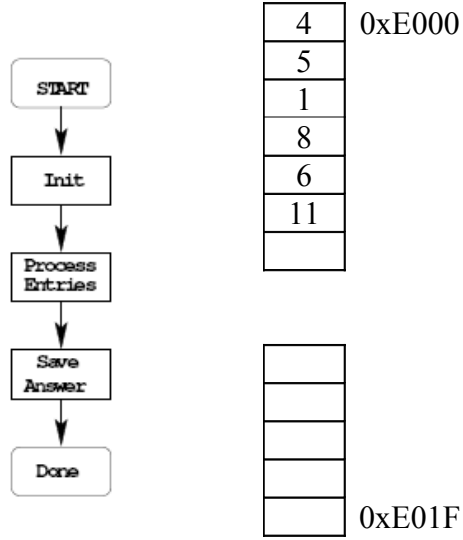
SUM ODD NUMBERS IN ARRAY FROM 0xE000 TO 0xE01f

Treat numbers as 8-bit unsigned numbers

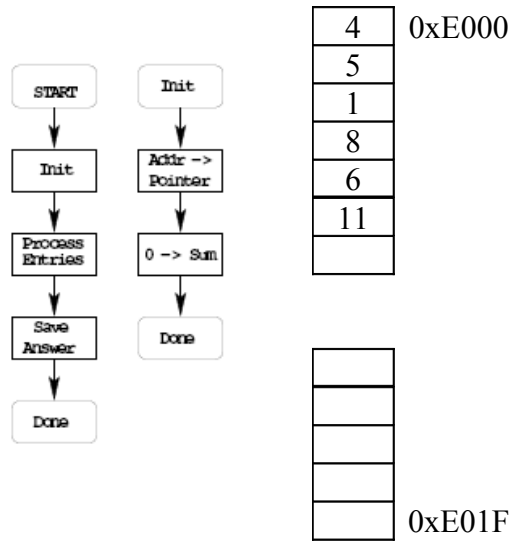
4	0xE000
5	
1	
8	
6	
11	

	0xE01F

### Start with the big picture



### Add details to blocks



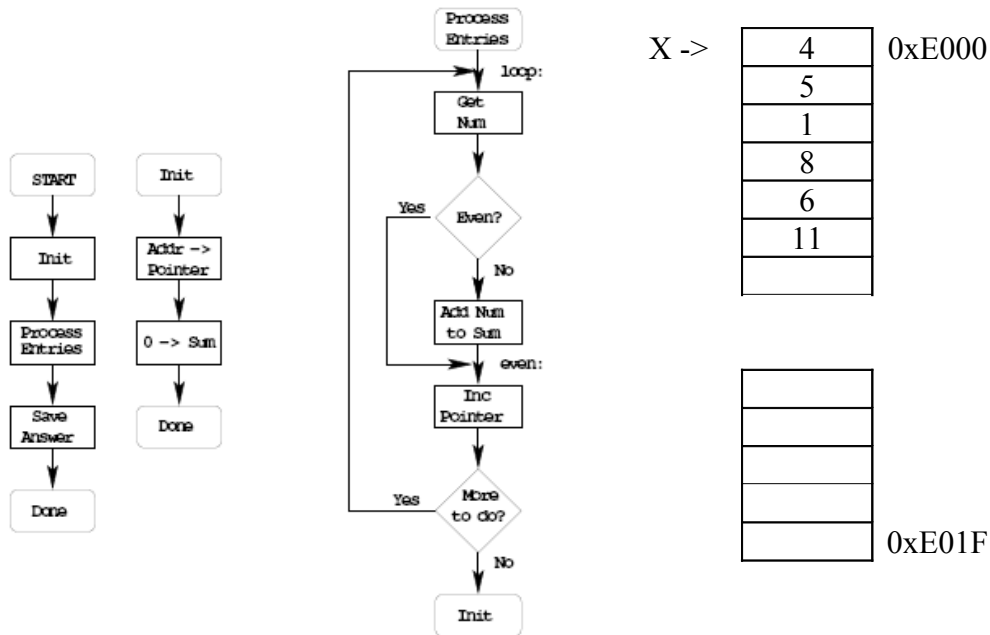
**Decide on how to use CPU registers for processing data**

**Pointer: X or Y -- use X**

**Sum: 16-bit register  
D or Y**

**No way to add 8-bit number to D  
Can use ABY to add 8-bit number to Y**

**Add more details: Expand another block**



**More details: How to tell if number is odd, how to tell when done**

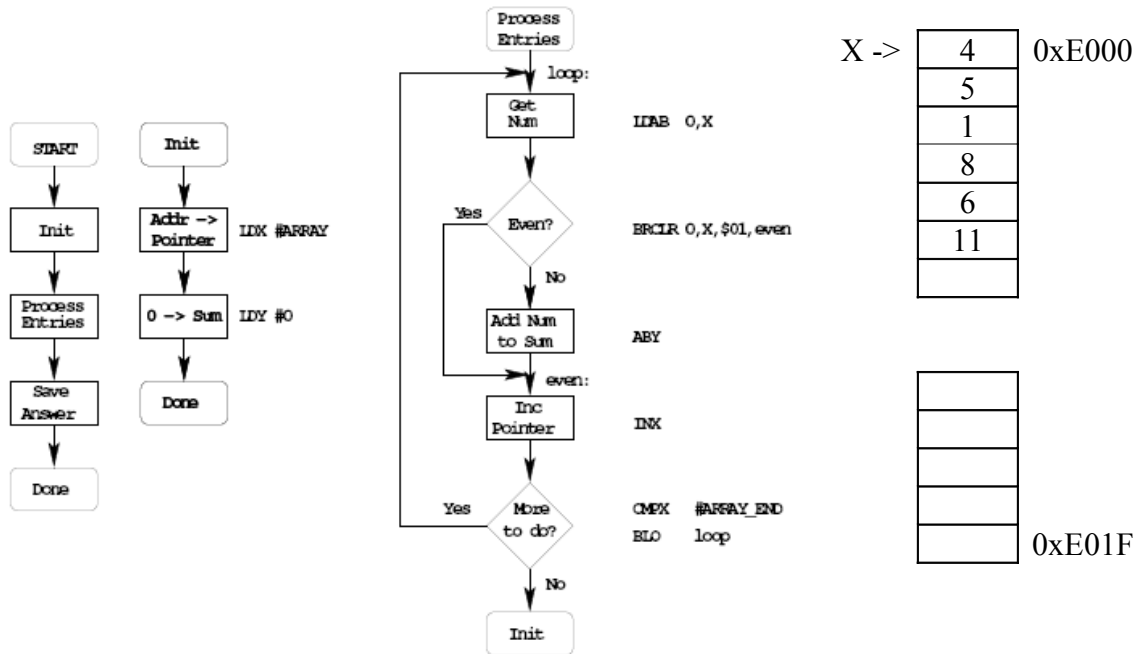
**How to test if even?  
LSB = 0 - check LSB of memory  
BRCLR 0,X,\$01,even**

**How to check if more to do?  
If X < 0xE020, more to do.  
CMPX #SE020  
BLO or BLT loop?**

**Address in unsigned, use unsigned compare  
BLO loop**

**Convert blocks to assembly code**





**How to test if even?**  
**LSB = 0 – check LSB of memory**  
**BRCLR 0,X,\$01,even**

**How to check if more to do?**  
**If X < 0xE020, more to do.**  
**BLO loop**

### Write program

;Program to sum odd numbers in a memory array

```

prog:    equ    $1000
data:    equ    $2000
array:   equ    $E000
len:     equ    $20

          org    prog
          ldx    #array      ; initialize pointer
          ldy    #0         ; initialize sum to 0
loop:    ldab   0,x         ; get number
          brclr  0,x,$01,skip ; skip if even
          aby    ; odd - add to sum
skip:    inx    ; point to next entry
  
```

```

cpx  #(array+len) ; more to process?
blo  loop          ; if so, process
sty  answer        ; done -- save answer
swi

```

```

org  data
answer: ds.w 1      ; reserve 16-bit word for answer

```

- Important: Comment program so it is easy to understand.

### The assembler output for the above program

- Note that the assembler output shows the op codes which the assembler generates for the HC12.
- For example, the op code for brclr 0,x,\$01,skip is 0f 00 01 02

as12, an absolute assembler for Motorola MCU's, version 1.2e

```

1000          prog: equ    $1000
2000          data: equ    $2000
e000          array:equ   $E000
0020          len:  equ    $20
1000          org      prog
1000 ce e0 00          ldx  #array      ; initialize pointer
1003 cd 00 00          ldy  #0          ; initialize sum to 0
1006 e6 00          loop: ldab  0,x      ; get number
1008 0f 00 01 02      brclr  0,x,$01,skip; skip if even
100c 19 ed          aby          ; odd - add to sum
100e 08          skip: inx          ; point to next entry
100f 8e e0 20          cpx  #(array+len); more to process?
1012 25 f2          blo  loop        ; if so, process
1014 7d 20 00          sty  answer    ; done -- save answer
1017 3f          swi

2000          org      data
2000          answer: ds.w 1          ; reserve 16-bit word
                                          ; for answer

```

```

Executed: Sun Jan 07 10:00:02 2007
Total cycles: 36, Total bytes: 24
Total errors: 0, Total warnings: 0

```

Here is the .s19 file:

```

S011000046696C653A2074657374332E730A76
S1131000CEE000CD0000E6000F00010219ED088ECD
S10B1010E02025F27D20003FE1
S9030000FC

```