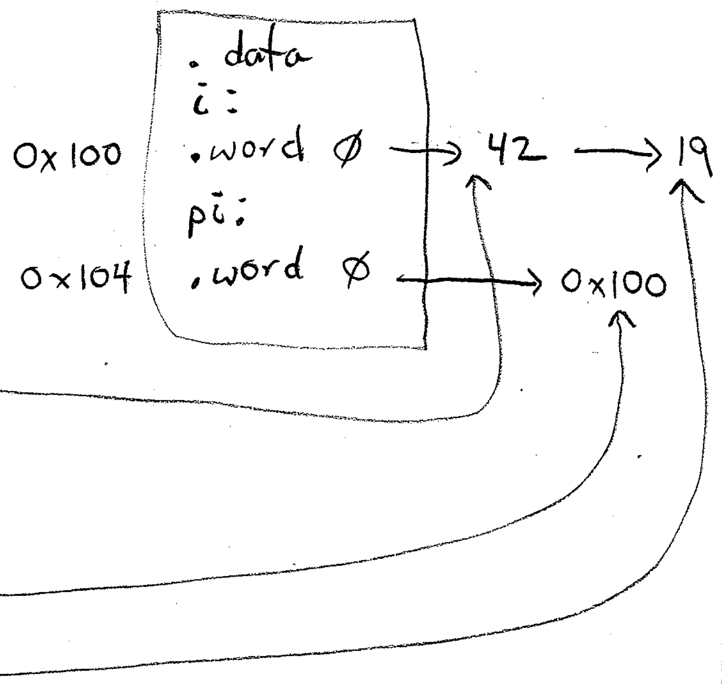
Pointers in C

key idea from yesterday : there are 2 ways to modify a variable in C
1) direct assignment
2) indirect using a pointer

```
int i;
int *pi;
```



1) direct assignment

```
i = 42;
```

2) indirect by pointer

```
pi = &i;
*pi = 19;
```

aside: `*pi = 19;` involves 2 memory accesses
(i) getting value of pi using <u>ldw</u> to retreive 0x100 from address 0x104
(ii) doing assignment *_ = using <u>stw</u> to address retreived in step (i)

`i = 42;` involves 1 memory access
(i) doing assignment _ = directly to ρ using <u>stw</u>.

When used with subroutines, we can change the value of a variable in a caller routine (eg, main()) in 2 ways:

1) return new value to caller, let it directly change the variable
2) caller sends pointer to variable as parameter, subroutine indirectly changes the variable

example:

1)
```
int getValue()
{
   return 42;
}
```

in main():
```
i = getValue();
```

2)
```
void getValueMod(int *pi)
{
      *pi = 19;
}
```

in main():
```
getValueMod(&i)
```

both change value of i

note: if you try this instead, it doesn't work:
```
getvalue(i);
```

you use this technique already with scanf()

```
/* Pointer example */

int   globalVariable;
int *pGlobalVariable;

int getValue()
{
        return 42;
}

void getValueByModifyingParam( unsigned int *pParam )
{
        *pParam = 19;
}

int main( int argc, char *argv[] )
{
        unsigned int i;
        i = getValue();                         // sets i=42

        pGlobalVariable = &globalVariable;      // computes address of globalVariable
        globalVariable = 19;                    // sets globalVariable = 19
        i = *pGlobalVariable;                   // sets i=19
        *pGlobalVariable = 42;                  // sets globalVariable = 42
        i = globalVariable;                     // sets i=42

        getValueByModifyingParam( &i );         // sets i=19
        return 0;
}
```

```c
/* Simple I/O example */

/*
 * This demo program is NOT intended for use with 259macros.h or 259library.c
 * Those files already defines some of these constants and functions.
 */

volatile unsigned int *pLEDR    = (unsigned int *)0x10000000;
volatile unsigned int *pLEDG    = (unsigned int *)0x10000010;
volatile unsigned int *pSWITCH  = (unsigned int *)0x10000040;
volatile unsigned int *pHEX7SEG = (unsigned int *)0x10000020;

#define DIGIT0  0x3F
#define DIGIT2  0x5B
#define DIGIT5  0x6D
#define DIGIT9  0x67

unsigned int getSwitch()
{
        return *pSWITCH;
}

void getSwitchParam( unsigned int *pParam )
{
        *pParam = *pSWITCH;
}

int main( int argc, char *argv[] )
{
        unsigned int msg, i=0, c;

        // display a message on 7-segment display
        msg = (DIGIT0<<24) | (DIGIT2<<16) | (DIGIT5<<8) | (DIGIT9<<0) ;
        *pHEX7SEG = msg;

        while( 1 ) {

                c = getSwitch();        // use either this style,
                // getSwitchParam( &c );        // or use this style

                *pLEDR = c;             // put switch pattern on LEDR

                i++;                    // increase counter i
                *pLEDG = (i >> 16);     // put bits 23..16 of counter i on LEDG
        }

        return 0;
}
```

```
/* Printing to terminal example */

/*
 * This demo program is NOT intended for use with 259macros.h or 259library.c
 * Those files already defines some of these constants and functions.
 */

#define ADDR_LEDG       0x10000010      /* output only, GREEN (DE1 8b, DE2 9b) */
#define ADDR_JTAG       0x10001000      /* send/recv chars to Terminal window */

volatile unsigned int *pLEDG            = (volatile unsigned int *)ADDR_LEDG;
volatile unsigned int *pTERMINAL        = (volatile unsigned int *)(ADDR_JTAG + 0);
volatile unsigned int *pTERMINALCTL     = (volatile unsigned int *)(ADDR_JTAG + 4);

void putcharJTAG( int c )
{
        int num_bytes_available;

        // wait until terminal is ready to accept another character
        do {
                num_bytes_available = (*pTERMINALCTL) >> 16;
        } while( num_bytes_available == 0 );

        *pTERMINAL = c;         // output the character
}

void printstringJTAG( char *psz )
{
        while( *psz ) {
                putcharJTAG( *psz++ );
        }
}

int getcharJTAG()
{
        unsigned int jtagword;
        int data_ready;

        // wait until terminal sends a character
        do {
                jtagword = *pTERMINAL;
                data_ready = (jtagword>>15) & 1;
        } while( data_ready == 0 );

        return jtagword & 0xff; // return the character
}

int main( int argc, char *argv[] )
{
        int c;
        char *pszMessage = "hello world.\n";

        printstringJTAG( pszMessage );

        while( 1 ) {
                c = getcharJTAG();
                putcharJTAG( c );       // put character on terminal
                *pLEDG = c;             // also display ASCII code on LEDG
        }

        return 0;
}
```