Take a Look Under the Hood

Embedded Systems Interfacing

1

Overview

- Strings
- Memory Space and Program Space
- MAP
- Character Arrays and Strings
- Pointers
- The Heap
- C-30 Memory Model

2

Strings

- Single character
  char ch;
  char ch=0x41;
  char ch='a';
- Array of characters
  char str[5]={'H','e','l','l','o'};
- Null terminated string
  char str[ ]={"Hello"};

C-30 calculates size of 6

C-30 adds '\0'

3

String Runtime Library

- #include <string.h>
- Memory copy, compare, move and initialize (set)
- String copy, concatenate, compare (collate), scan, tokenize, length and error message strings
- Some functions have safe versions available
  - Specify string length
  - Does not just depend on '\0'
- See web

4

Memory Space Allocation

- char s[ ]="Hello world!";
  - C-30 linker reserves contiguous memory in RAM (data space) which is part of ndata (near data section).
  - C-30 linker stores inialized value in long table (in program memory) which is part of init code section.
  - C-30 compiler creates routine to copy from init to ndata section which is part of c0 code.

Use twice the space

5

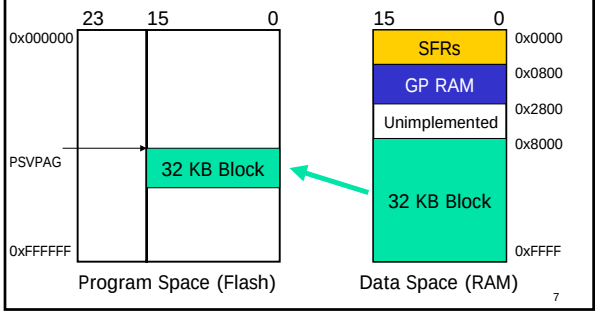
Memory Space Allocation

- const char s[ ]="Hello world!";
  - C-30 linker stores initialized value in long table (in program memory) which is part of init code section.
  - Can not copy other string over top of this string because of const.

Use less space

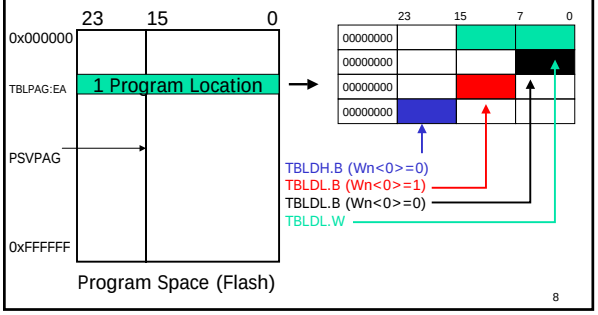
6

Program Space Visibility



7

Program Space Visibility



8

Program Space Visibility

- PSV is used to manage constant arrays (numeric of string) so that a single type of pointer to the data memory bus Can be used uniformly for constants and variables
- Table access mechanism is used to perform variable initialization, limited to c0- segments, for maximum compactness and efficency

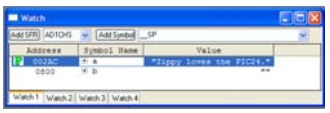
9

Investigating Memory Allocation

```
#include <string.h>

const char a[]="Zippy loves the PIC24.";
char b[100];

int main(){
    strncpy(b,"MPLAB C-30",sizeof(b));
    //strncpy is safer than Di Jasio's code
    return (0);
}
```

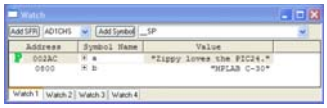
10

Watch Window



Before Execution

After Execution



11

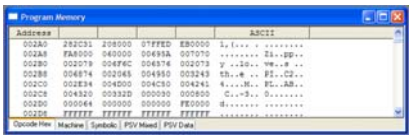
Disassembled Listing

Map File

```
Program Memory Usage
Section address    length (PC units)   length (bytes) (dec)
------- -------    -----------------   --------------------
.reset          0               0x4            0x6     (6)
.ivt            0x4             0xfc           0x17a  (378)
.aivt           0x104           0xfc           0x17a  (378)
.text           0x200           0xac           0x102  (258)
.const          0x2ac           0x22           0x33    (51)
.dinit          0x2ce           0x8            0xc     (12)
.isr            0x2d6           0x2            0x3     (3)
__CONFIG2       0x157fc 0x2                    0x3     (3)
__CONFIG1       0x157fe 0x2                    0x3     (3)
   Total program memory used (bytes):           0x444  (1092) <1%
Data Memory Usage
Section address      alignment gaps    total length  (dec)
------- -------      --------------    -------------------
.nbss   0x800          0               0x64  (100)
     Total data memory used (bytes):          0x64  (100) 1%
```

13

.const in Program Memory



14

Pointers

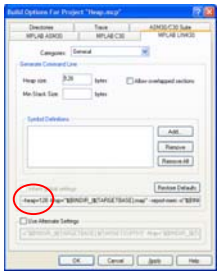
sizeof(a)/sizeof(int) = 20/2=10

```
int main(){
     int *pi;        // pointer to integer
     int i;          // index or counter
     int a[10];      // array of ten integers

     for(i=0;i<sizeof(a)/sizeof(int);i++)          //safer than Di Jasio's code
          a[i]=i;    // access array with index

     pi=a;           // initialize array pointer
     for(i=0;i<sizeof(a)/sizeof(int);i++,pi++)     //safer than Di Jasio's code
          *pi=-i;    //access array with pointer

     return (0);
}
```

15

The Heap

- Heap is open unused RAM
- Allocate via command line argument for linker
- Allocate large area so malloc( ) and calloc( ) can make most efficient use of heap



16

Code Using Heap

```
#include <stdlib.h>
#define ARRAY_SIZE 10

int main(){
     int * pi;          // pointer to integer array
     int * pi_end;      // end of array
     int i;             // index or counter

     pi=(int *)malloc(ARRAY_SIZE*sizeof(int));  //allocate space for array in heap
     pi_end=pi+ARRAY_SIZE;                      //specify byte past end of array
     for(i=0;pi<pi_end;i++,pi++)
          *pi=i;                                // access array with pointer into heap
     free((void *)pi);
     return (0);
}
```

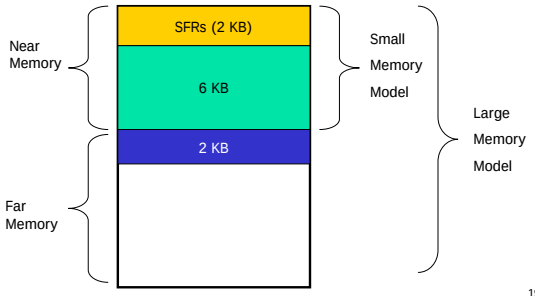
17

C-30 Memory Model

- Near memory is first 8KB
  - 2 KB SFRs
  - 6 KB RAM
  - Local and global variables in Near memory
- Rest of RAM is Far memory
  - Uses indirect addressing
  - Stack access indirectly and in Far memory
  - Heap accessed indirectly (via pointer) and may be in Far memory

18

C-30 Memory Model



19

References

- Books
  - Textbook -- Chapter 6
  - 16-Bit Language Tools Library[1]
  - MPLAB C30 C Compiler User's Guide[2]

[1] DS51456 [2] DS51284

20

Homework

- Chapter 6
  - 1 (String search in array of strings]
    Use the strstr( ) in string.h
  - 2 (Binary Search of array of integers)
    Assume binarySearch( ) is passed number, pointer to array, and length of array. It returns an index for number equal to number in array. If number is not in array, the return value will be index of next lowest number. Assume array is sorted from smallest to largest.

22