On-board RAM  |  External RAM

0x0100  |  0x0000  |  0x04FF  |  0x0050  |  0xFFFF  |  0x0FFF

.data variables  |  .bss variables  |  HEAP  |  STACK  |  __data_start

__data_end == __bss_start
__bss_end
]*( __malloc_heap_start) == __heap_start
*( __brkval) (<= *SP - *( __malloc_margin))
__data_end == __bss_start
*( __malloc_heap_start) == __heap_start

CMPE-311 C Programming & Embedded Systems
UMBC-CE 2014
Global & Static Variables

On-board RAM

0x0100

.data variables

__data_start

0x04FF

.bss variables

__bss_end

0x0500

HEAP

.*(__malloc_heap_start) == __heap_start

0xFFFF

STACK

.External RAM

.*(__brkval) (<= SP - *(__malloc_margin))

__bss_end

.*(__malloc_heap_start) == __heap_start

__data_end == __bss_start
Memory Sections Tips

Global & Static Variables

On-board RAM

External RAM

0x0100
0x04FF
0x0500
0xFFFF

Dynamic Allocation (Grows towards RIGHT)

*(__brkval) <= *SP - *(__malloc_margin))

*(__malloc_heap_start) == __heap_start

__data_start

__data_end == __bss_start

__bss_end

SP

RAMEND
Memory Sections Tips

Global & Static Variables

On-board RAM

External RAM

0x0100

0x04FF

0x0500

0xFFFF

Dynamic Allocation (Grows towards RIGHT)

__data_start

.data variables

.bss variables

HEAP

STACK

SP

RAMEND

*(__brkval) <= *SP - *(__malloc_margin))

__bss_end

*(__malloc_heap_start) == __heap_start

__data_end == __bss_start

Local variables, stack variables such as return addresses
(Grows towards LEFT and can collide with HEAP, .data or .bss)
Tips for using program space

▶ What happens when you run out of memory?
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
- Shorten your prompts (reuse parts of your string)
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
- Shorten your prompts (reuse parts of your string)
- Don’t have multiple temporary string variables in memory at the same time
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
- Shorten your prompts (reuse parts of your string)
- Don’t have multiple temporary string variables in memory at the same time
- Have gcc optimize for space
  (include \texttt{-Os} option during compilation)
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
- Shorten your prompts (reuse parts of your string)
- Don’t have multiple temporary string variables in memory at the same time
- Have gcc optimize for space
  (include -Os option during compilation)
- Check your stack pointer

  printf("sp:%d\n",SP);
Tips for using program space

- What happens when you run out of memory?
- Data should be below 1K bytes
- Shorten your prompts (reuse parts of your string)
- Don’t have multiple temporary string variables in memory at the same time
- Have gcc optimize for space (include -Os option during compilation)
- Check your stack pointer
  
  ```
  printf("sp:%d\n",SP);
  ```
- Documentation for memory sections available from [here](#)
Using program space

- AVR has a library for keeping const data in program memory (flash) and accessing it directly instead of using RAM
Using program space

- AVR has a library for keeping const data in program memory (flash) and accessing it directly instead of using RAM
- Program Memory has 16KB space
Using program space

- AVR has a library for keeping const data in program memory (flash) and accessing it directly instead of using RAM.
- Program Memory has 16KB space.
- Example functions available in stdio.h:
  - `printf_P`
  - `sprintf_P`
  - `fprintf_P`
  - `fputs_P`
  - `fscanf_P`
Using program space

- #include <avr/pgmspace.h>
Using program space

- #include <avr/pgmspace.h>

- Declare const strings using special flag PROGMEM

  //PROGMEM used to locate a variable in flash ROM

  const PROGMEM char myString[] = "Repeated Use";
Using program space

- `#include <avr/pgmspace.h>

- Declare const strings using special flag PROGMEM

  ```
  //PROGMEM used to locate a variable in flash ROM
  const PROGMEM char myString[] = "Repeated Use";
  ```

- You can create special pointers using PGM_P and access the data using a special macro `pgm_read_byte`
Using program space

- `#include <avr/pgmspace.h>`

- Declare `const` strings using special flag PROGMEM

  ```
  const PROGMEM char myString[] = "Repeated Use";
  ```

- You can create special pointers using `PGM_P` and access the data using a special macro `pgm_read_byte`

- Documentation is available from [here](#)
Examples

PGM_P progPtr;
progPtr = myString;

//Somewhere in your code

while(pgm_read_byte(progPtr)! = '\0'){
    printf("%c", pgm_read_byte(progPtr));
    progPtr++;
}

#include <avr/pgmspace.h>

void lcd_puts_P(const char c[]) { 
//same const char *c
uint8_t ch = pgm_read_byte(c);
while(ch != 0) {
lcd_putchar(ch);
ch = pgm_read_byte(++c);
}
}

// Usage: Note PSTR macro which simplifies placing string
// literals in flash ROM
// Code: lcd_puts_P(PSTR("Hello World"));
// Or: const PROGMEM char SOME_STRING[] = "Repeated Use";
//      lcd_puts_P(SOME_STRING);