UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

11032 GNU

Hacking the GNU Linker

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip
UofM
MASTERs 2007

Class Objective

When you finish this class you will:

- Understand compiler optimizations and how they might affect your code
- Understand the link30 process
- Understand *gld* files and how they affect your final code
- Be able to modify linker scripts

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip
U of M
MASTERs 2007

Agenda

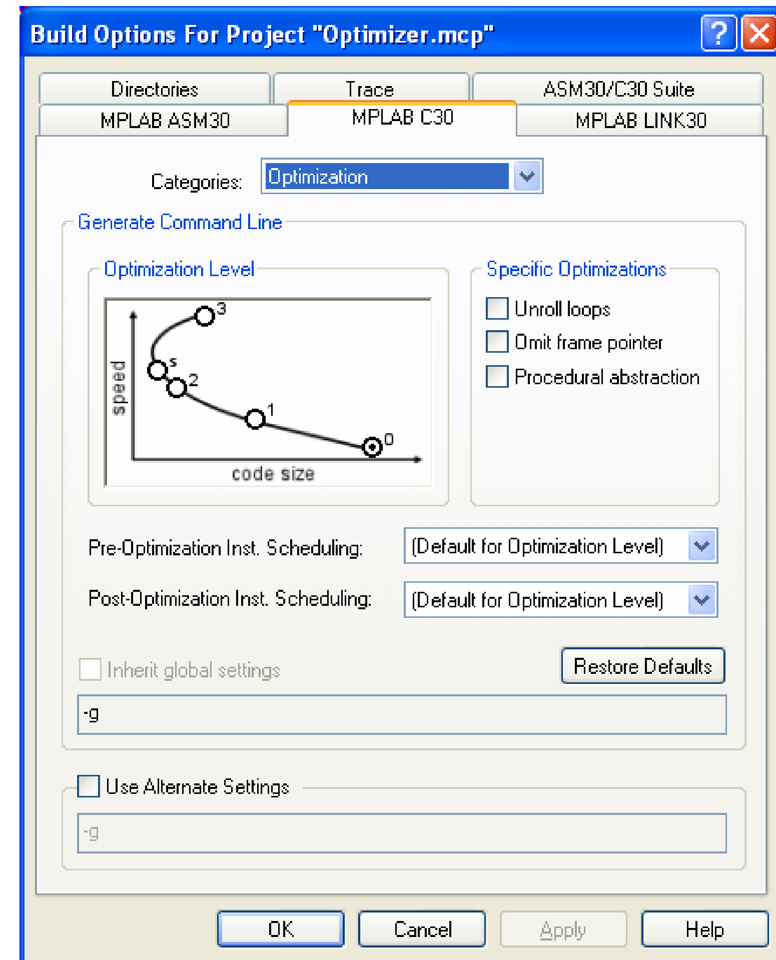
- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

C30 Optimizer

4 levels of optimization

- **-O0 None**
  - Default Level
- **-O1**
  - Try to reduce code size and execution time
    - **If conversion**
    - **Merge constants**
    - **Dead code elimination**
    - **Redundancy analysis**



© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip UofM MASTERs 2007

C30 Optimizer continued

- **-O2**
  - Optimize yet more
    - **Crossjumping (weak PA)**
    - **Regmove**
    - **Takes longer to evaluate but produces smaller and faster code**
- **-O3**
  - All of the above plus
    - **Inline-functions**
    - **Move invariant branches out of loops**
    - **Note that this does not unroll-loops**
- **-Os**
  - Optimize for speed
    - **Most O2 switches**
    - **Reorder blocks in functions to reduce branches**

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Example of Optimization

```
const unsigned int xadder = 3;
const unsigned int yadder = 3;
int main(void)
{
   unsigned int x, y, z, i;
   // set the bit as an output
   TRISAbits.TRISA0 = 0;
   x = 0;        y = 10;
   for(i = 0; i < 20; i++) {
      x = x + xadder;
      y = y + yadder;
      z = 42;
      LATAbits.LATA0 = !LATAbits.LATA0;
   }
   LATA = x;              // here to ensure values are used after
   LATA = y;              // otherwise they will be optimised out
   LATA = z + 1;
   while (1);
}
```

Level	Size FLASH/RAM	
-O0	321/8	Vars on stack R-M-W
-O1	255/2	x+y+z eval as const, reg tied
-O2	255/2	
-O3	255/0	Fully reg tied
-Os	249/0	
-O3 –funroll-loops	441/0	Unrolled!

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip
U of M
MASTERs 2007

Agenda

- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip
UofM
MASTERs 2007

Undetected Coding Errors

or… what breaks the compiler!

- **Shared variables that should be volatile <u>not</u> marked as volatile**
  - With no optimization the compiler will usually save the results of calculations back to memory
  - O1 and above will try and keep commonly used variables in registers
  - An ISR using the variable will load its own value from memory and store back to memory however the interrupted function will still be using a value cached in a register

© 2007 Microchip Technology Incorporated. All Rights Reserved.

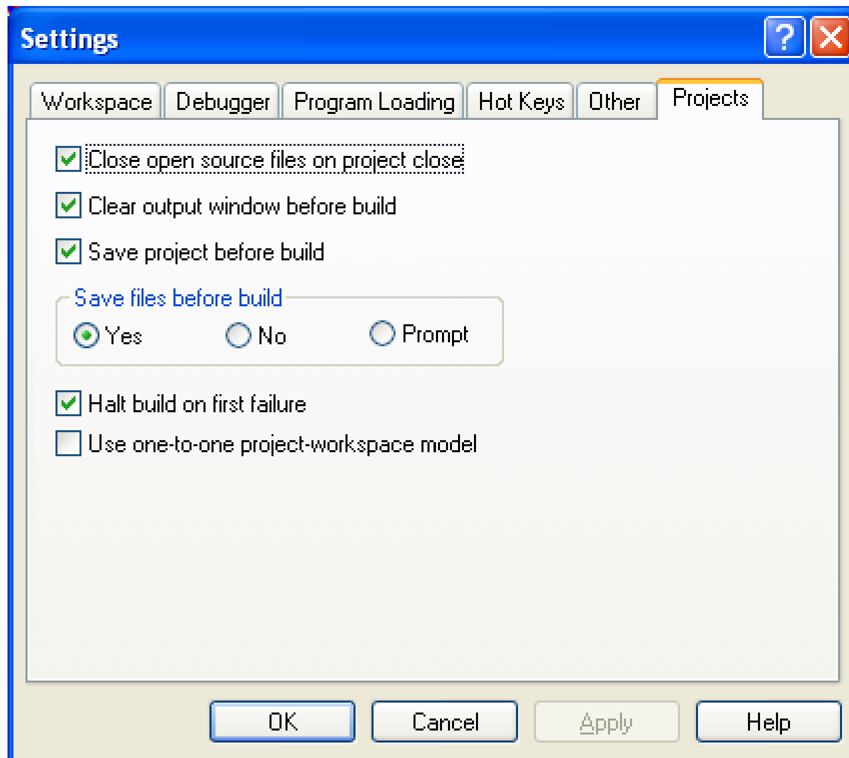
Lab 1

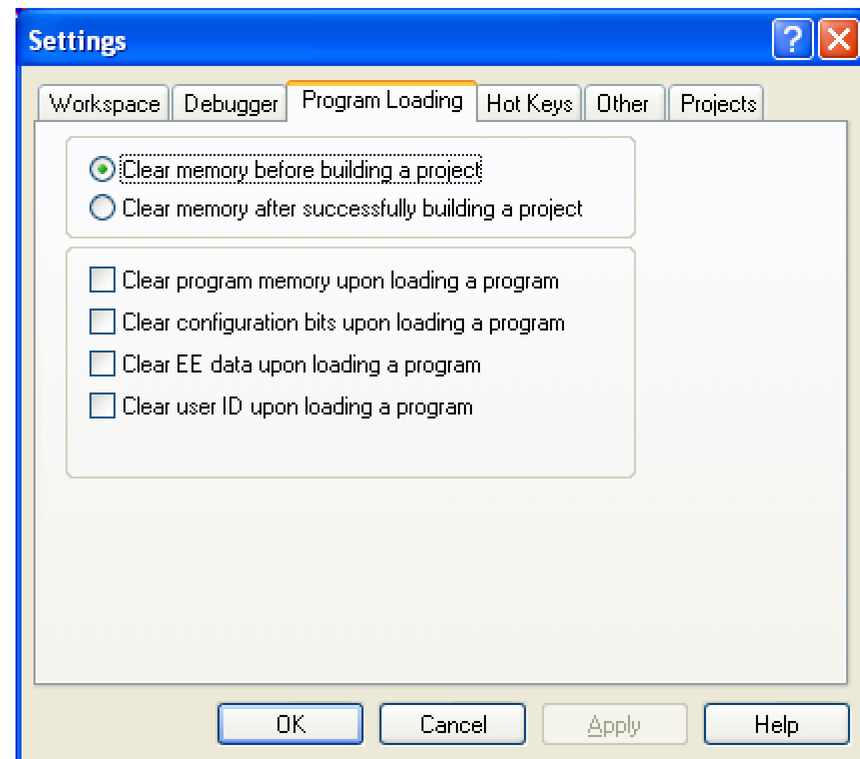
- **Tasks**
  - Set up MPLAB® IDE for later
  - See what optimization does
  - See what MPLAB C30 does

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP UOFM MASTERs 2007

Hands-On!





© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UofM
MASTERs 2007

Hands-On!

- **Example code**
  - Load up and run C:\Masters\11032\optimizer*optimizer.mcw*
    - **Check the output on the logic analyser**
  - Turn on optimisation and observe the output
    - **Add the volatile keyword and run**
- **Also try**
  - C:\Masters\11032\simplescript*simplelink.mcw*
  - Check the output
    - **Add the commented line back in and run it**
    - **What is the problem?**

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP U OF M MASTERS 2007

More undetected coding errors

- **Inline assembly**
  - Simple *asm* statements work okay at –O0
    - **Assumptions made about register and memory use may not work with optimization turned on**
    - **Use extended asm statements**
      - asm ("swap %0" : "+r"(var));
      - Let the compiler know what resources you are using and what you clobber
  - Attempting Control Flow
    - **Do not use hardware *do* loops**
    - **Move them to a separate .s file and wrap the function properly**
- **Be kind to your compiler**
  - And it will be kind to you

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP U OF M MASTERs 2007

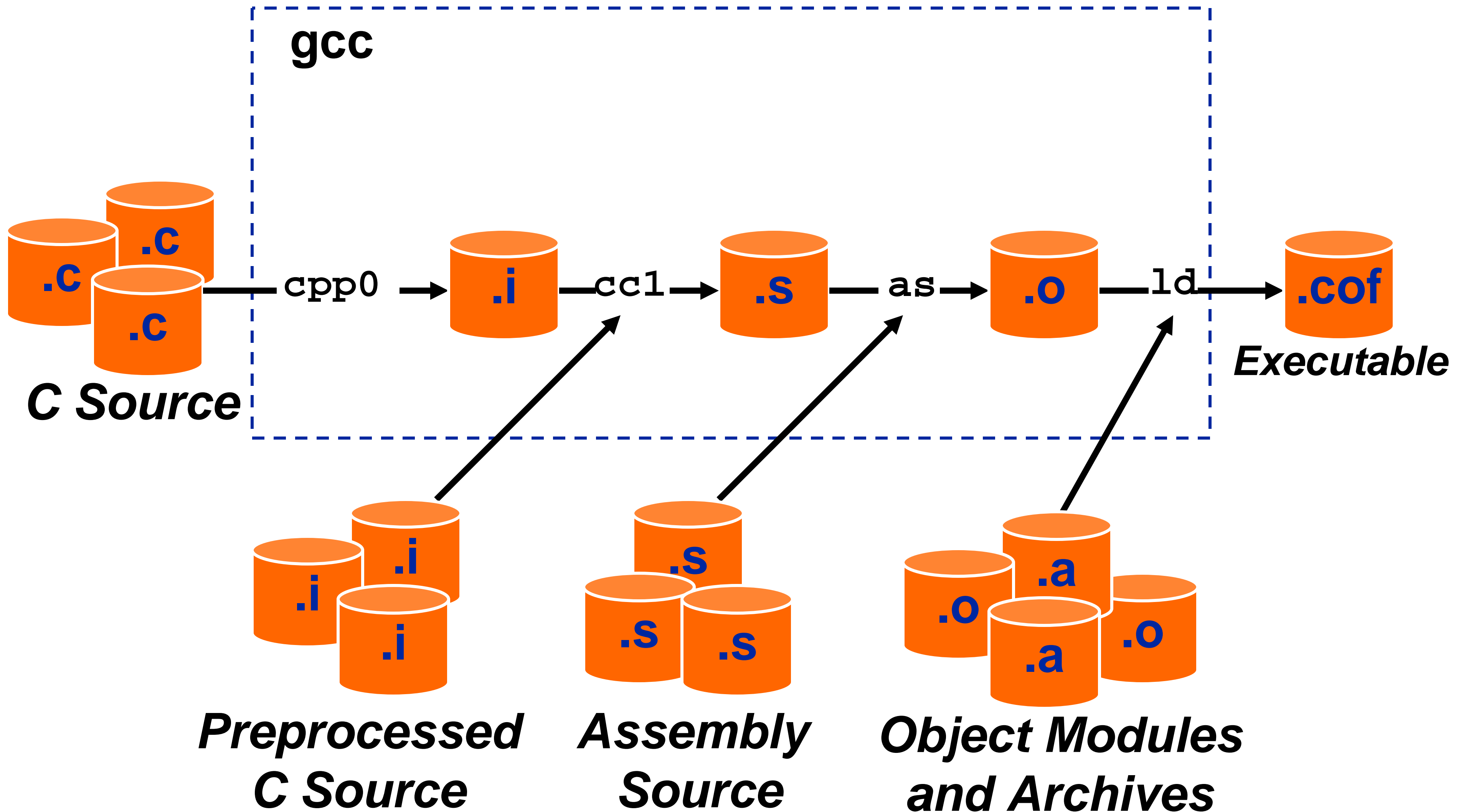
Agenda

- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UofM
MASTERs 2007

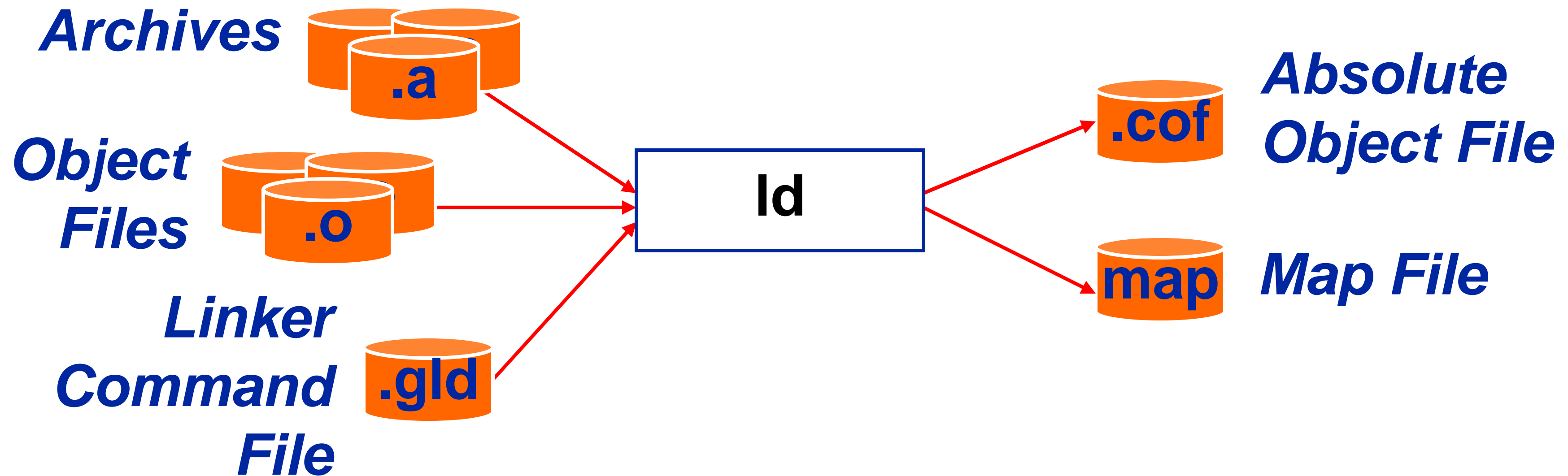
GNU Build Process



© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip UofM MASTERs 2007

Linker Overview



- Assigns absolute addresses and resolves symbol references
- Command file describes memory organization and logical section location
- Maps input sections from the code to output sections in the hex file

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

GNU Linkage Steps

- **Loading Input Files**
  - Check format, Build Tables of Sections
- **Allocating Memory**
  - Generate output sections from input sections
- **Resolving Symbols**
  - Convert symbols into section relative values
  - Match up all externally referenced symbols
- **Creating Special Sections**
  - Such as .handle, and .dinit
- **Computing Absolute Addresses**
  - Convert all section relative addresses into absolutes
- **Building the Output File**

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip
U of M
MASTERs 2007

Agenda

- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Linker Scripts *.gld* Files

- Processor Family
- Memory Regions
- Base Addresses
- Reset Section
- Input to Output Section Maps
- Interrupt Vector Tables
- SFR Addresses

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

Terminology

Section Name	Attribute	Comment
.text	code	Program code
.data	data	Initialized data
.bss	bss	Un-initialized data, (Block Started by Symbol)
.xbss, .ybss	bss, xmemory	Un-initialized data in x/y memory
.xdata, .ydata	data, xmemory	Initialized data in x/y memory
.nbss	bss, near	near memory, excludes X
.ndata	data, near	
.ndconst	data, near	Constants, but in RAM
.pbss	bss, persist	Persistent memory, un-initialized, not zeroed
.dconst	data	
.const	psv	Constant data in PSV space
.eedata	eedata	Data in EEPROM

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

Simple Example

- **With the previous hands-on example still open, look at the *gld* file**
  - Cut down version of normal file
  - All unused definitions removed
  - Note the interrupt section where padding has been used to ensure correct alignment

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP UOFM MASTERs 2007

Memory Regions

- Declare regions of memory that the linker can use to place code and data
- Flags used to help best-fit algorithm

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Memory Regions

```
data   (a!xr) : ORIGIN = 0x800,        LENGTH = 0x7800
reset         : ORIGIN = 0x0,          LENGTH = 0x4
ivt           : ORIGIN = 0x4,          LENGTH = 0xFC
program (xr)  : ORIGIN = 0x400,        LENGTH = 0x2A800
CONFIG1       : ORIGIN = 0xF80000,     LENGTH = 0x2
CONFIG2       : ORIGIN = 0xF80002,     LENGTH = 0x2
FGS           : ORIGIN = 0xF80004,     LENGTH = 0x2
FOSCSEL       : ORIGIN = 0xF80006,     LENGTH = 0x2
FOSC          : ORIGIN = 0xF80008,     LENGTH = 0x2
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Reset Section

- **Constructs the *goto* instruction at location 0x000000**
- **__*reset* symbol defined in *crt0.s***
  - initialize SP and SPLIM
  - initialize PSV if __const_length > 0
  - process the .dinit template
    - **or link with *crt1.s***
  - call _main

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Reset Section

```
.reset :
 {
        SHORT(ABSOLUTE(__reset));
        SHORT(0x04);
        SHORT((ABSOLUTE(__reset) >> 16) & 0x7F);
        SHORT(0);
 } >reset
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

Section Maps

- **Input Sections defined in the application**

```
void filterFunction(int num, int* src);
unsigned char strData[40];
int coefficients[] = { 4, 5, 29, 34};
int __attribute__((section(".mySection")))
   filterData[34];
```

- **Converted to Output Sections in the COF file**
  - Defined by the linker and *gld* file
  - Grouped

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

Section Map

```
.text __CODE_BASE :
{
      *(.init);
      *(.handle);
      *(.libc) *(.libm) *(.libdsp);
      *(.lib*);
      *(.text);
} >program
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Interrupt Vector Tables

- **Laid out to match the devices *IVT* & *AIVT***
  - One entry per interrupt vector slot
- **C like notation to call user function or default**

Interrupt Vector Table

```
.ivt __IVT_BASE :
{
        ..
        ..
        LONG( DEFINED(__OscillatorFail) ? ABSOLUTE(__OscillatorFail) :
                ABSOLUTE(__DefaultInterrupt));

        ..
        ..
        LONG( DEFINED(__T2Interrupt) ? ABSOLUTE(__T2Interrupt) :
                ABSOLUTE(__DefaultInterrupt));
  } >ivt
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

SFR Names

- **SFR names allow symbols to be resolved**
- **Two forms**
  - Without '_' for assembly language code
  - With leading '_' to comply with names that the C compiler generates

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip UofM MASTERs 2007

Agenda

- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Things to do in the linker

- Fix the stack at a specific location and/or length
- Calculate the length of a section or module
- Use one handler for multiple interrupts
- Build a Bootloader

© 2007 Microchip Technology Incorporated. All Rights Reserved.

University of Microchip UofM MASTERs 2007

Stack Size and Allocation

- **Stack is located at the bottom of RAM**
  - Linker defines __*SP_init*
- **Allocated largest available block**
  - Linker defines __*SPLIM_init*

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Stack Size and Allocation

```
.stack 0x1800 :
{
        __SP_init = .;
        . += 0x100;
        __SPLIM_init = .;
        . += 8;
} >data
```

Section 11.11
ASM30/LINK30
manual

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

C Equivalent

- **It is also possible to define a stack in C**
  - Use symbol names *_SP_init* and *_SPLIM_init*
  - Fix their locations using the address attribute
  - Must not be initialised by crt0.s
  - Note the guardband

© 2007 Microchip Technology Incorporated. All Rights Reserved.

C Stack allocation

```
int __attribute__((address(0x1800), persistent)) _SP_init[128];
int __attribute__((address(0x1900), persistent)) _SPLIM_init[4];

int main(void)
{
        unsigned int x;
        ..
        ..
}
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

Length of Sections

- **A Bootloader may need to know the length of a module or section**
  - Used for CRC checking the app
  - Only calculated at link time
  - Stored along with the program code

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

Length of Sections

```
.text __CODE_BASE :
{
        LONG(SIZEOF(.text));
        LONG(.tEnd - .tStart);
        *(.init);
         *(.handle);
         *(.libc) *(.libm) *(.libdsp); *(.lib*);
         .tStart = . ;
         *(.text);
         .tEnd = . ;
} >program
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP UOFM MASTERs 2007

Common Interrupt Handler

- **One interrupt handler can handle multiple interrupts:**

```
void __attribute__((interrupt)) _CommonHandler(void)
{
        // do something then clear interrupt sources
        IFS0bits.T2IF = IFS0bits.T3IF = 0;
}
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Common Interrupt Handler

```
.ivt __IVT_BASE :
{
  ..
  LONG( DEFINED(__T2Interrupt) ? ABSOLUTE(__T2Interrupt) :
         ABSOLUTE(__DefaultInterrupt));
  LONG( DEFINED(__T3Interrupt) ? ABSOLUTE(__T3Interrupt) :
         ABSOLUTE(__DefaultInterrupt));
} >ivt
```

```
.ivt __IVT_BASE :
{
  ..
  LONG( DEFINED(__CommonHandler) ? ABSOLUTE(__CommonHandler) :
         ABSOLUTE(__DefaultInterrupt));
  LONG( DEFINED(__CommonHandler) ? ABSOLUTE(__CommonHandler) :
         ABSOLUTE(__DefaultInterrupt));
} >ivt
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

Common Interrupts in C

```
// declare the T3 Interrupt routine and alias it to T2 ISR
void __attribute__((interrupt, auto_psv, alias("_T2Interrupt")))
        _T3Interrupt(void);


// the T2 ISR handler
void __attribute__((interrupt, auto_psv)) _T2Interrupt(void)
{
        // toggle the state
        bState = !bState;
        IFS0bits.T2IF = IFS0bits.T3IF = 0;
}
```

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Agenda

- Optimization in the GNU compiler
- Common coding pitfalls
- The GNU linker Steps
- Contents of linker scripts
- Changes to linker scripts
- A complex application

© 2007 Microchip Technology Incorporated. All Rights Reserved.

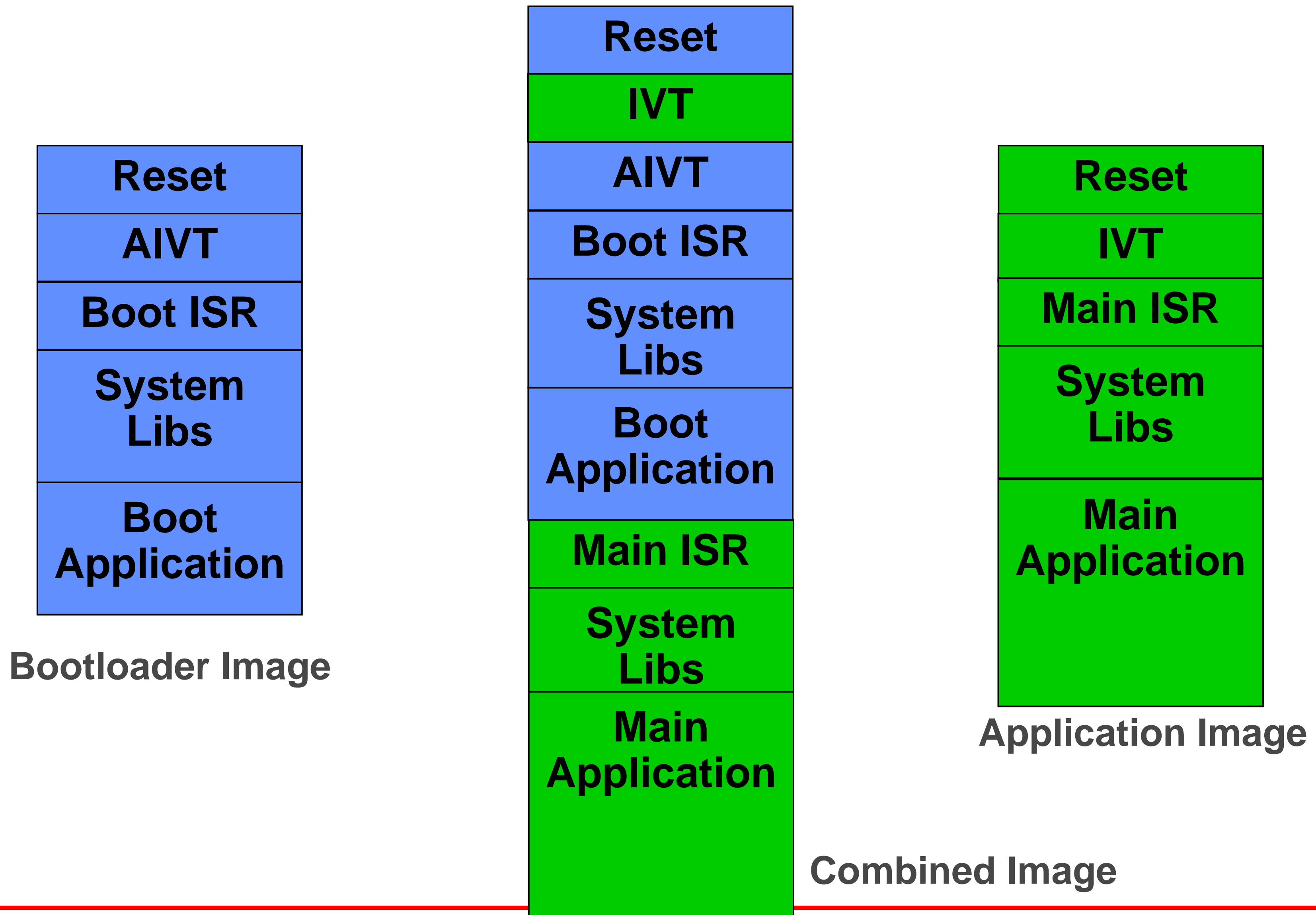
University of Microchip U of M MASTERs 2007

Bootloaders

- **For 16-bit, Bootloaders are common**
  - AN851, Examples for all 16-bit devices
- **Frequently, customer specific needs**
  - Special memory layout
  - Common functions (3DES, TCP/IP)
- **What changes are needed to gld files?**
  - Only examples, customers will differ
- **How to demonstrate and test Bootloaders?**
  - Running two applications in MPLAB® IDE is possible

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Bootloader Memory Layout



© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

Task Outline

- Restrict the Bootloader to a limited section of memory
- Define the application address range
- Make the Bootloader call the main code
- Handle interrupt sources in the Bootloader and application
- Prevent the application overwriting the reset section and aivt tables
- Prove it works in the simulator!

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

Hands-On Tasks

- **In the Bootloader**
  - Modify the gld file so that the program section starts at 0x200 and is 0x200 in length
  - Change the __CODE_BASE to 0x200
  - Verify that the gld only has an aivt section and no ivt section
  - Define a function pointer in bootloader.c and set it to the main application entry point (0x400)
  - Switch in the aivt whilst waiting for a timeout
  - Switch out the aivt and call the main application

- **In the main application**
  - Verify in the gld file that the program section starts at 0x400 and is 0x2A800 long
  - Comment out the .reset output section
  - Verify that the gld only has an ivt section and no aivt section

- **In MPLAB® IDE**
  - Ensure multiple project views are enabled
  - Settings>Program Loading
    - **Clear all tick boxes**
  - Set Bootloader as active project
  - Make application (using right-click)
  - Make Bootloader
  - Debug->Animate

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Suggestions for Linker Files

- **Modify one item at a time**
  - Test and retest
  - Customers have most success by just modifying output sections
- **The linker is extremely fragile**
  - Try to use C attributes whenever possible
  - Trying to be too clever will break the linker

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Summary

- **Compiler optimizations affect code density and speed**
  - They can have side effects on your code
- **The linker is a powerful way of laying out your application in memory**
  - Many modifications can be done in ‘C’
  - But some can only be done at the link stage
- ***gld* Files define how the linker works**
  - Complex applications may require *gld* changes
  - Modify these files with care

© 2007 Microchip Technology Incorporated. All Rights Reserved.

Any Questions?

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

References

- MPLAB® C30 C Compiler User's Guide – DS51284
- MPLAB ASM30, MPLAB LINK30 and Utilities User's Guide – DS51317

© 2007 Microchip Technology Incorporated. All Rights Reserved.

UNIVERSITY OF MICROCHIP
UOFM
MASTERs 2007

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLoq, KeeLoq logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.