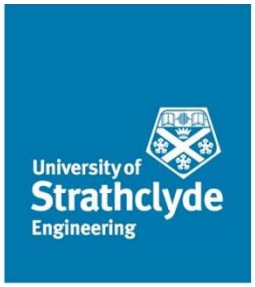
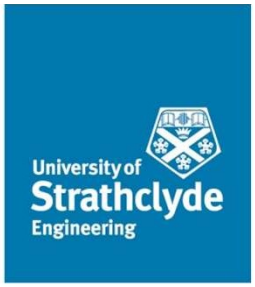


Working with assembler

James Irvine

DEPARTMENT OF ELECTRONIC & ELECTRICAL ENGINEERING

Assembler Baby Steps



- Write the program in C
- Add an assembler module to the project
- Write a routine in assembler
- Call it from C
  - Remember to declare it as extern in the C module
  - Declare it as public in assember

Simple C program

```
#include "msp430.h"

int main( void )
{
  int counter;
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // enables the gpio on the 4133
  // PM5CTL0 &=~LOCKLPM5;
  P1DIR = 0x01;
  while (1)
  {
    if (counter++ < 0) P1OUT = 0x00;
    else P1OUT = 0x01;
  }
}
```

Add some assembler

- Create a new assembler file setup.asm
- Add setup.asm to the project
  - Project → Add files
- Start with an empty file, make sure it compiles and runs
- Then start moving over code

Simple C program

```
#include "io430.h"

extern void setup();

int main( void )
{
  int counter;
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // enables the gpio for the 4133
  // PM5CTL0 &=~LOCKLPM5;
  setup();
  P1DIR = 0x01;
  while (1)
  {
    if (counter++ < 0) P1OUT = 0x00;
    else P1OUT = 0x01;
  }
}
```

Initial asm routine file

```
#include "msp430.h"                ; #define controlled include file

        NAME    setup                   ; module name

        PUBLIC  setup                   ; make the setup label visible
                                        ; outside this module

        RSEG    CODE                    ; place routine in 'CODE' segment

setup:  NOP                             ; do nothing
        RET                             ; then return


END
```

Start transferring code

```
#include "msp430.h"                ; #define controlled include file

        NAME    setup                   ; module name

        PUBLIC  setup                   ; make the main label visible
                                        ; outside this module

        RSEG    CODE                    ; place routine in 'CODE' segment

setup:  MOV.B   #0x1, P1DIR             ; Set DDR for P1.0 output
        RET                             ; then return


END
```

Updated C program

```
#include "io430.h"

extern void setup();

int main( void )
{
  int counter;
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // enables the gpio for the 4133
  // PM5CTL0 &=~LOCKLPM5;
  setup();
  while (1)
  {
    if (counter++ < 0) P1OUT = 0x00;
    else P1OUT = 0x01;
  }
}
```

More transferring

```
#include "msp430.h"                ; #define controlled include file

        NAME    setup

        PUBLIC  setup                   ; make the setup label visible
                                        ; outside this module
        PUBLIC  flash                   ; and make flash visible outside

        EXTERN  counter                 ; bring in the counter variable from c

        RSEG    CODE                    ; place routine in 'CODE' segment

setup:  MOV.B   #0x1, P1DIR             ; Set DDR for P1.0 output
        RET                             ; then return

flash:  NOP                             ; Do nothing
        RET                             ; then return

END
```

Updated C program

```
#include "io430.h"

extern void setup();
int counter;          // counter now a global variable so the assembler can see it

int main( void )
{
// Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // enables the gpio for the 4133
  // PM5CTL0 &=~LOCKLPM5;
  // initialize the IO
  setup();
  while (1)
  {
    flash();
    if (counter++ < 0) P1OUT = 0x00;
    else P1OUT = 0x01;
  }
}
```

Move the functionality

```
#include "msp430.h"                ; #define controlled include file

        NAME    setup

        PUBLIC  setup              ; make the setup label visible
                                   ; outside this module
        PUBLIC  flash              ; and make flash visible outside

        EXTERN  counter            ; bring in the counter variable from c

        RSEG    CODE               ; place routine in 'CODE' segment

setup:  MOV.B   #0x1, P1DIR        ; Set DDR for P1.0 output
        RET                        ; then return

flash:  INC.W counter
        JN led_on                  ; is incremented counter negative?
        MOV.B   #0x1,P1OUT         ; if not, turn LED off
        RET                        ; then return
led_on: MOV.B   #0x0,P1OUT         ; if so, turn it on
        RET                        ; then return

        END
```

Updated C program

```
#include "io430.h"

extern void setup();
int counter;

int main( void )
{
// Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // enables the gpio for the 4133
  // PM5CTL0 &=~LOCKLPM5;
  // initialize the IO
  setup();
  while (1)
  {
    flash();
  }
}
```

Conclusions

- This very simple example could have been done in assembler from the start
- Most large programs use assembler selectively
  - For speed
  - For memory efficiency
  - For control
- Getting things working in C, then optimizing modules using assembler is a good approach

University of Strathclyde Glasgow

DEPARTMENT OF ELECTRONIC & ELECTRICAL ENGINEERING