EECS 373
Design of Microprocessor-Based Systems

Robert Dick
University of Michigan

Lecture 3: Linking, debugging

13 September 2017

Many slides from Mark Brehob

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)
xPSR

Review

- ISA
  - Encodings
  - Addressing modes
  - Status register
  - Using the ARM ARM
- ABI (including quick rules
  - Pass in r0-r3
  - Return in r0 (+r1)
  - Caller saved r0-r3
  - Callee saved r4-r7
  - Others? See more detailed ABI information
- Build process
  - Gcc, nm, objdump, as, ld
  - Make and makefiles
  - More on this today

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

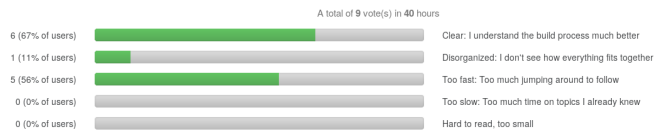
Lab 2

Why did that happen?
- Disable watchdog
- Odd target address for bx, blx
  - Long history of such features

Survey outcome

In-class examples closes in 8 hours

A total of **9** vote(s) in **40** hours



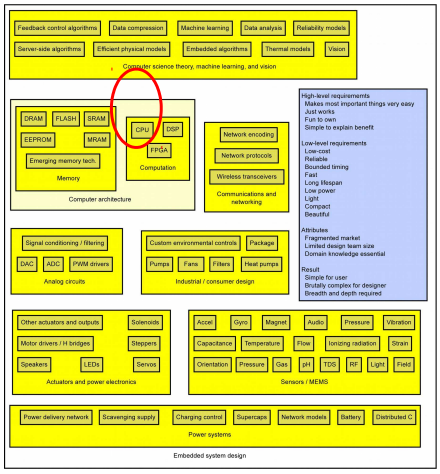
Examples have value
They are covered too fast
Could better show how things fit together

Resolution
- Narrate what I am doing and why more thoroughly
- Slow down

Outline

- Where are we?
- Building and linking
- Debugging

An embedded system



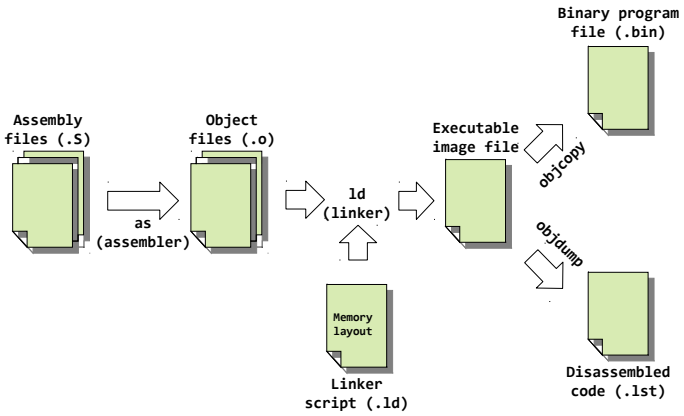
Outline

- ~~Where are we?~~
- Building and linking
- Debugging

What are the real GNU executable names for the ARM?

- Just add the prefix “arm-none-eabi-” prefix
- Assembler (as)
  - arm-none-eabi-as
- Linker (ld)
  - arm-none-eabi-ld
- Object copy (objcopy)
  - arm-none-eabi-objcopy
- Object dump (objdump)
  - arm-none-eabi-objdump
- C Compiler (gcc)
  - arm-none-eabi-gcc
- C++ Compiler (g++)
  - arm-none-eabi-g++

How does an assembly language program get turned into a executable program image?



What information does the disassembled file provide?

```
all:
    arm-none-eabi-as -mcpu=cortex-m3 -mthumb example1.s -o example1.o
    arm-none-eabi-ld -Ttext 0x0 -o example1.out example1.o
    arm-none-eabi-objcopy -Obinary example1.out example1.bin
    arm-none-eabi-objdump -S example1.out > example1.lst
```

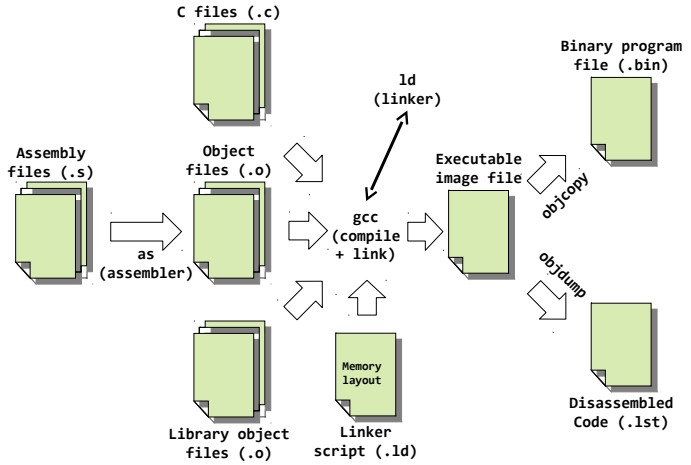
```
    .equ  STACK_TOP, 0x20000800
    .text
    .syntax    unified
    .thumb
    .global    _start
    .type start, %function

_start:
    .word STACK_TOP, start
start:
    movs r0, #10
    movs r1, #0
loop:
    adds r1, r0
    subs r0, #1
    bne  loop
deadloop:
    b    deadloop
    .end
```

```
example1.out:     file format elf32-littlearm

Disassembly of section .text:

00000000 <_start>:
   0: 20000800   .word 0x20000800
   4: 00000000   .word 0x00000000

00000008 <start>:
   0: 200a       movs  r0, #10
   2: 2100       movs  r1, #0

00000004 <loop>:
   4: 1809       adds  r1, r1, r0
   6: 3801       subs  r0, #1
   8: d1fc       bne.n c <loop>

0000000a <deadloop>:
   a: e7fe       b.n   12 <deadloop>
```

Elements of assembly language program?

```
    .equSTACK_TOP, 0x20000800    /* Equates symbol to value */
    .text                        /* Tells AS to assemble region */
    .syntax unified              /* Means language is ARM UAL */
    .thumb                       /* Means ARM ISA is Thumb */
    .global _start               /* .global exposes symbol */
                                 /* _start label is the beginning */
                                 /* ...of the program region */
    .type   start, %function     /* Specifies start is a function */
                                 /* start label is reset handler */
_start:
    .word   STACK_TOP, start     /* Inserts word 0x20000800 */
                                 /* Inserts word (start) */
start:
    movs r0, #10                 /* We’ve seen the rest ... */
    movs r1, #0
loop:
    adds r1, r0
    subs r0, #1
    bne  loop
deadloop:
    b    deadloop
    .end
```

How does a mixed C/Assembly program get turned into a executable program image?



Outline

- ~~Where are we?~~
- ~~Building and linking~~
- Debugging

Compile-time debugging

- -Wall: Show more compile-time problems
- -ggdb: Include the most complete debugging information possible.
- -O0: Turn off optimization (only when debugging).

What do debuggers do?

- Souce→PC association
- Breakpoints
  - Single stepping
  - Skip counts
- Variable inspection
  - Monitoring
  - Stack analysis
  - Memory search
  - Setting variables
- Backtracing

What is about to happen?

- Try each of these on real example
- Use same debugger you use in class, but w.o. GUI wrapper
- Show the commonly used debugging functions

Done.