Embedded Systems Programming

ES Development Environment (Module 3)

Yann-Hang Lee

Arizona State University
yhlee@asu.edu
(480) 727-7507

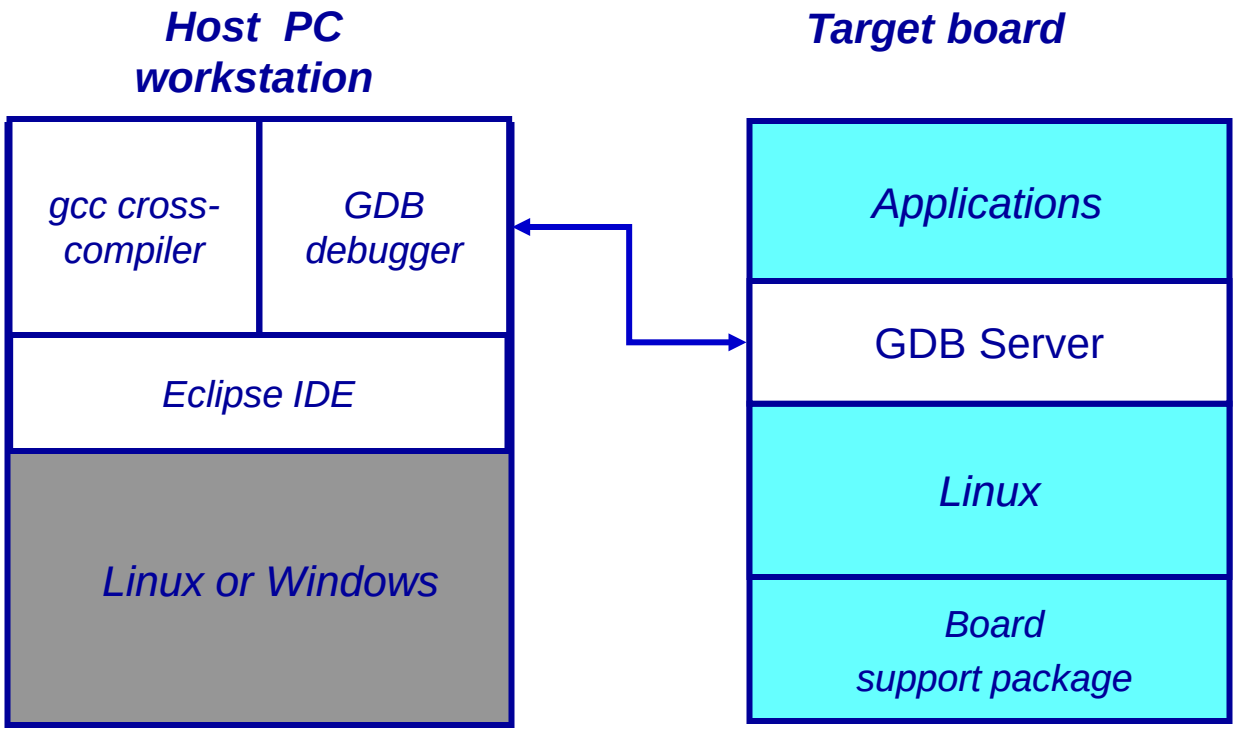
Summer 2014

Embedded System Development

- **Need a real-time (embedded) operating system (RTOS) or run on bare metal ?**
- **Need a development and test environment ?**
  - Use the host to edit, compile, and build application programs
  - At the target, use tools to load, execute, debug, and monitor (performance and timing)

Development Host (gnu tool chain + Eclipse) ⟷ *Target* ⟷ *Peripherals + Plant Model*

Development Environment of Linux



Command line or IDE (Eclipse, Kdevelop)

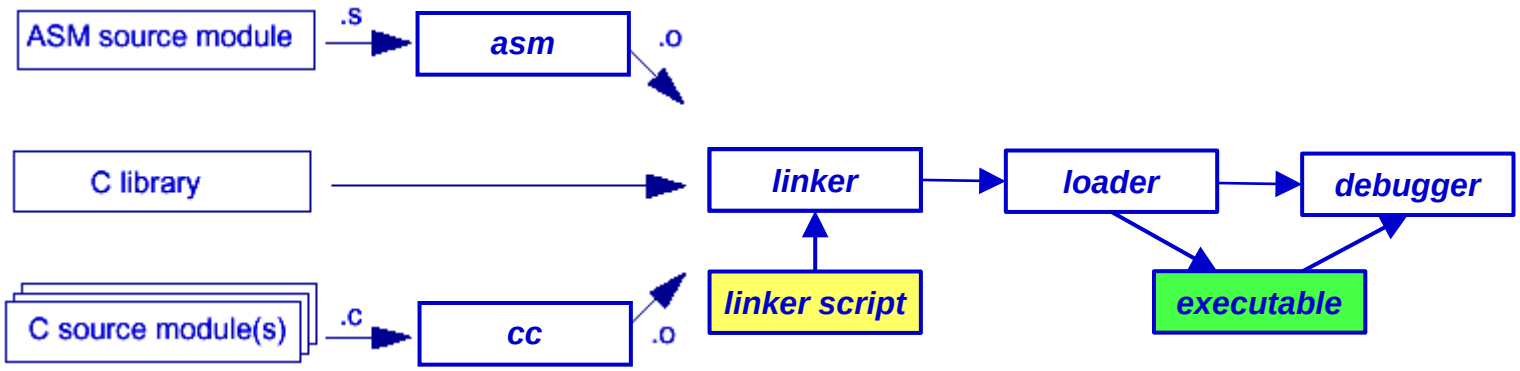
Linux and GNU tools

HelloWorld Example

```
/* Hello World program */
#include <stdio.h>
#include <time.h>

int main(void) {
    int i;
    time_t   curTime;

    for (i=0; i<10; i++)  {
            curTime = time(NULL);
            printf("Hello, world. The current time is: %s", ctime (&curTime));
            sleep(i);
    }
}

CC = i586-poky-linux-gcc
all: HelloWorld.c
            $(CC)   -o HelloWorld.o   HelloWorld.c

scp   HelloWorld.o   root@10.218.101.25:/home/root/labs/
```

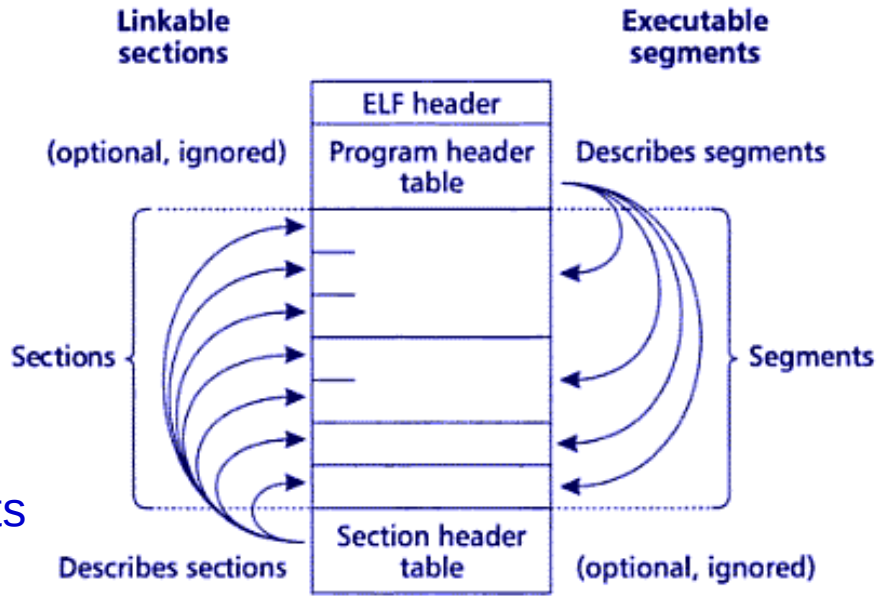
From Source to Executable

- Compiler, linker, and loader
- In ELF: executable, relocatable, shared library, and core
  - information for relocation, symbol, debugging
  - linker resolves symbol reference
- Link script or link command file
  - assigns absolute memory addresses (program area, static data, bss, stack, vector table, etc.)
- Startup code to disable interrupts, initialize stack, data, zero uninitialized data area, and call main().



ELF -- Executable and Linking Format

- **The old one – a.out for UNIX community**
- **ELF –**
  - standard for Linux
  - better support for cross-compilation, dynamic linking, initializer/finalizer
- **An ELF file can be one of the 4 types**
  - Relocatable
  - Executable
  - Shared object
  - Core file
- **Two views**
  - Compilers, assemblers, and linkers – a set of logical sections.
  - System loader – a set of segments



Linker Script

- **A text file that describe how the sections in the input files should be mapped into the output file, and to control the memory layout of the output file.**
  - ENTRY linker script command to set the entry point.
  - SECTIONS command to describe memory layout
  - `.' indicates the location counter
  - MEMORY command describes the location and size of blocks of memory in the target

```
MEMORY
{
        ROM (rx) : ORIGIN = 0, LENGTH = 256k
        RAM (wx) : org = 0x00100000, len = 1M
}
```

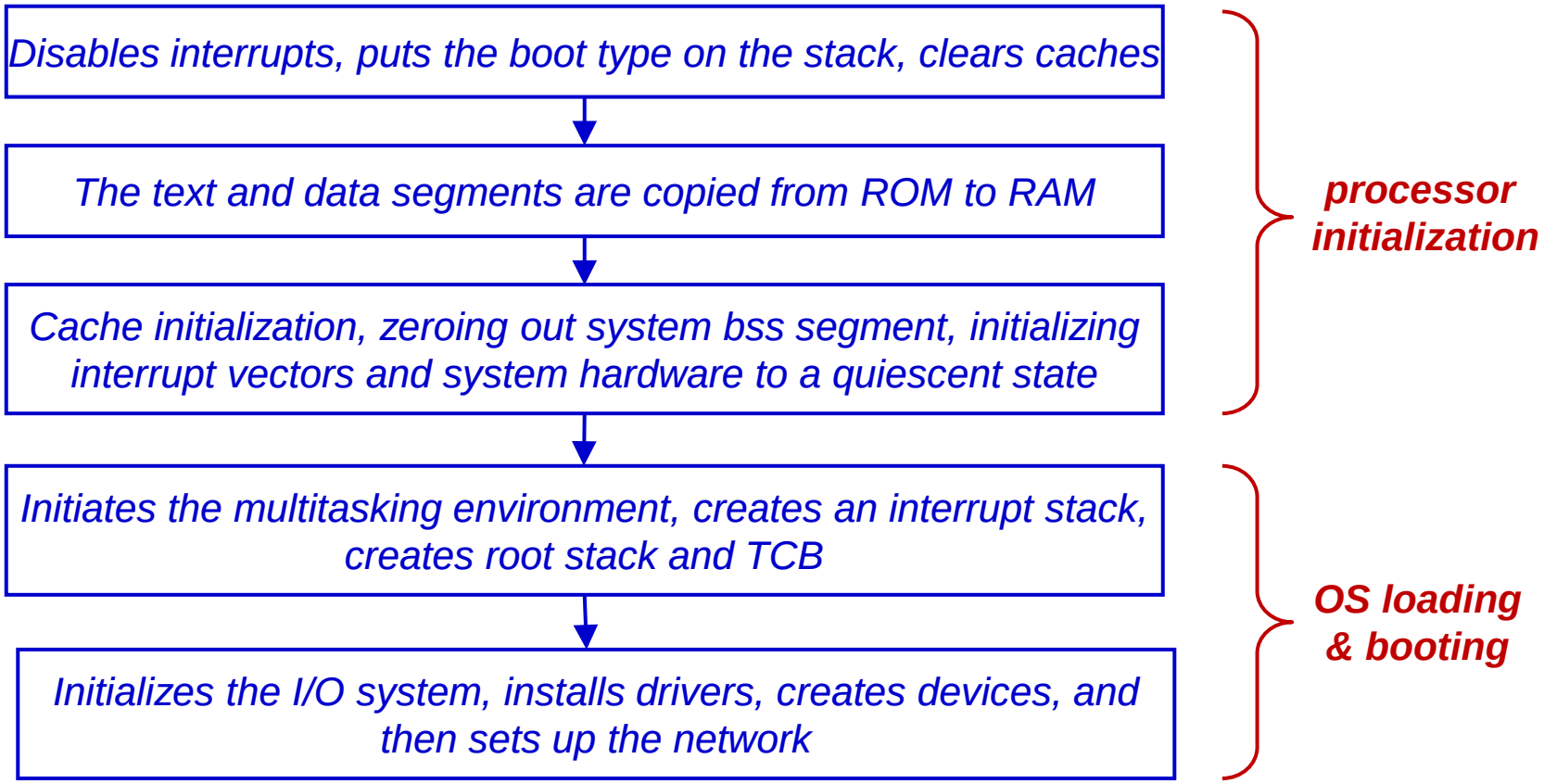
```
SECTIONS
{
        .text.start (_KERNEL_BASE_) : {
              startup.o( .text )
        }

        .text : ALIGN(0x1000) {
              _TEXT_START_ = .;
              *(.text)
              _TEXT_END_ = .;
        }

        .data : ALIGN(0x1000) {
              _DATA_START_ = .;
              *(.data)
              _DATA_END_ = .;
         }

        .bss : ALIGN(0x1000) {
              _BSS_START_ = .;
              *(.bss)
              _BSS_END_ = .;
         }
}
```

Debugger for Embedded Systems

- **Observability, real-time analysis, and run control**
- **At host: GUI, source code, symbol table, type information, line number**
- **Communication to the target**
  - serial port, Ethernet, USB, JTAG, etc.
  - read/write memory and registers
  - execution control (single step, breakpoint, watchpoint, etc.)
- **At target:**
  - debugging stub (software): at breakpoint, replace the instruction with breakpoint inst. or invalid instruction to trigger exception
  - (USB) JTAG interface: hardware breakpoint, trace buffer, flash programming

Start-up and Booting Sequence

- Processor, memory, and IO initialization
- Load an operating system (or an application)



Supplementary Slides

ELF -- Executable and Linking Format (2)

- **Partial list of the ELF sections**
  - .init - Startup
  - .text - String
  - .fini - Shutdown
  - .rodata - Read Only
  - .data - Initialized Data
  - .tdata - Initialized Thread Data
  - .tbss - Uninitialized Thread Data
  - .ctors - Constructors
  - .dtors - Destructors
  - .got - Global Offset Table
  - .bss - Uninitialized Data