C and Embedded Systems

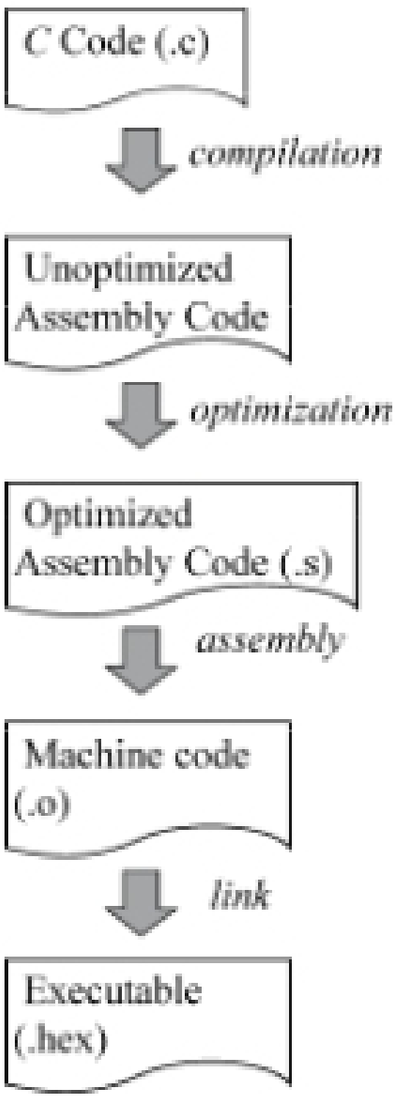
- A μC-based system used in a device (i.e., a car engine) performing control and monitoring functions is referred to as an **embedded system**.
  - The embedded system is invisible to the user
  - The user only indirectly interacts with the embedded system by using the device that contains the μC
- Many programs for embedded systems are written in C
  - Portable – code can be retargeted to different processors
  - Clarity – C is easier to understand than assembly
  - Modern compilers produce code that is close to manually-tweaked assembly language in both code size and performance

So Why Learn Assembly Language?

- The way that C is written can impact assembly language size and performance
  - i.e., if the **uint32** data type is used where **uint8** would suffice, both performance and code size will suffer.
- Learning the assembly language, architecture of the target µC provides performance and code size clues for compiled C
  - Does the µC have support for multiply/divide?
  - Can the µC shift only one position each shift or multiple positions? (i.e, does it have a *barrel shifter*?)
  - How much internal RAM does the µC have?
  - Does the µC have floating point support?
- Sometimes have to write assembly code for performance reasons.

C Compilation

From .c to .hex



Example Optimization

```
i = i + j;
k = k + j;
```

compilation

```
mov    j,W0   ;W0 = j
add    i      ;i = i + W0 = i + j
mov    j,W0   ;W0 = j
add    k      ;k = k + W0 = k + j
```

optimization

```
mov    j,W0   ;W0 = j
add    i      ;i = i + W0 = i + j
add    k      ;k = k + W0 = k + j
```

W0 already contains `j`,
remove second **`mov`** instruction