ECE 271 – Microcomputer Architecture and Applications Lecture 9

Vince Weaver

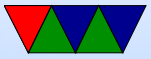
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

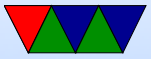
19 February 2019

Announcements

- Read Chapter 3.5 to 3.7

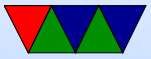


Lab #4 Pre-lab Notes



Assembler – Code comments

- Can use C and C++ style comments
- Keil: can use ; (makes rest of line a comment)
- Linux/gas: Can use @ for beginning of line



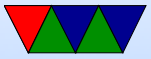
Setting peripheral registers in assembly

```
ldr r1,=RCC_BASE                ; r1=&RCC;
ldr r3,[r1,#RCC_AHB2ENR]        ; r3=RCC->AHB2ENR;
orr r3,r3,#RCC_AHB2ENR_GPIOBEN  ; r3=r3 | RCC_AHB3ENR_GPIOBEN;
str r3,[r1,#RCC_AHB3ENR]        ; RCC_>AHB2ENR=r3;
```

- The various constants are set in the provided .s files
- use the `BIC` instruction to clear bits

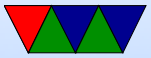
  Why not just use `AND`? Because a mask has lots of 1s

```
    and r3,#0xfffffffe      ; won't␣fit␣in␣instruction,␣too␣big
␣␣␣␣bic␣r3,#1␣␣␣␣␣␣␣␣␣␣␣␣;␣same␣as␣"and␣r3,#~1"␣but␣fits␣in␣constant
\end{lstlsiting}
␣␣␣␣\item␣You␣can␣include␣complicated␣C-stye␣constant␣manipulations,␣things␣like
\begin{lstlisting}
␣␣␣␣and␣r3,r3,#(GPIOBEN␣+␣0x5␣|␣(1<<3))
```



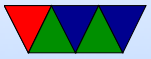
Something-cool Notes

- Using the BSRR register to set/reset bits without having to do a read/modify/write. Write a bit pattern 0 or 1, 0 means leave alone, 1 means set in the bottom 16 bits, clear in the top 16 bits.



(digression) Demoscene Assembly Language Demo

- Some people code assembly for fun
- Can even win fabulous cash prizes
- Including 1st-place retro demo from 2018 Demosplash

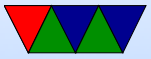


Disassembler

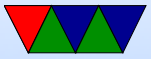
```
#include <stdio.h>
int i;
int main(int argc, char **argv) {
        for(i=0;i<100;i++) {
                printf("Hello!\n");
        }
        return 0;
}
```

```
gcc -Wall -mthumb -march=armv7-a -o test test.c
```

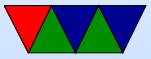
```
Disassembly of section .bss:

0002102c <i>:
```



```
   2102c:       00000000        andeq   r0, r0, r0

Disassembly of section .rodata:

000104fc <_IO_stdin_used>:
   104fc:       00020001        andeq   r0, r2, r1
   10500:       6c6c6548        cfstr64vs       mvdx6, [ip], #-288      ; 0xfffffee0
   10504:       Address 0x00010504 is out of bounds.

0001043c <main>:
   1043c:       b580            push    {r7, lr}
   1043e:       b082            sub     sp, #8
   10440:       af00            add     r7, sp, #0
   10442:       6078            str     r0, [r7, #4]
   10444:       6039            str     r1, [r7, #0]
   10446:       f241 032c       movw    r3, #4140       ; 0x102c
   1044a:       f2c0 0302       movt    r3, #2
   1044e:       2200            movs    r2, #0
   10450:       601a            str     r2, [r3, #0]
   10452:       e010            b.n     10476 <main+0x3a>
   10454:       f240 5000       movw    r0, #1280       ; 0x500
   10458:       f2c0 0001       movt    r0, #1
   1045c:       f7ff ef42       blx     102e4 <puts@plt>
   10460:       f241 032c       movw    r3, #4140       ; 0x102c
```

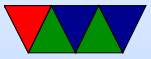


```
10464:      f2c0 0302       movt    r3, #2
10468:      681b            ldr     r3, [r3, #0]
1046a:      1c5a            adds    r2, r3, #1
1046c:      f241 032c       movw    r3, #4140       ; 0x102c
10470:      f2c0 0302       movt    r3, #2
10474:      601a            str     r2, [r3, #0]
10476:      f241 032c       movw    r3, #4140       ; 0x102c
1047a:      f2c0 0302       movt    r3, #2
1047e:      681b            ldr     r3, [r3, #0]
10480:      2b63            cmp     r3, #99 ; 0x63
10482:      dde7            ble.n   10454 <main+0x18>
10484:      2300            movs    r3, #0
10486:      4618            mov     r0, r3
10488:      3708            adds    r7, #8
1048a:      46bd            mov     sp, r7
1048c:      bd80            pop     {r7, pc}
```

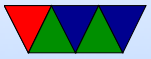


Assembler directives

- AREA – declare a new area (code/data/bss)
  `AREA myData, DATA, READWRITE`
- ENTRY – declare entry point into the code
  You might think this is "main" on C, but actually it is usually somthing called _start, a lot of things happen before main() gets called
- ALIGN – align the current memory address
  for performance, or some things (like variables on stack) must be aligned



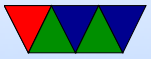
- DCB – reserve space for bytes
  array DCB 1,2,3,4
  hello DCB "Hello World!",0
- DCW – reserve space for 16-bit values
- DCD – reserve space for 32-bit values
- DCFS/DCFB – floating point
- SPACE – restore unreserved data (BSS)
  p SPACE 255 ; reserve 255 zeros
- FILL – reserve space and fill it with a value
  f FILL 20,0xff,1 ; allocate 20 bytes, fill with 0xff
- EQU – like #define in C, lets you set constants.



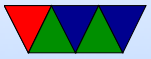
MAXCPUS EQU 8

On Linux same, but `.equ MAXCPUS, 8`

- RN – alias a register name, if you want to use something like X instead of R3
- EXPORT/IMPORT – export says to make symbol globally visible (.globl on Linux). Import says a symbol is external, like "extern" on C.
- INCLUDE/GET – like include directive in C, includes another file when assembling
- PROC/ENDP – start and end of function. Mostly to make debugging easier? Doesn't actually change



generated code?



Do another example if time

- Register allocation