ECE 498 – Linux Assembly Language Lecture 6

Vince Weaver

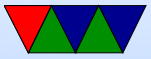
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

4 December 2012

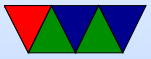
THUMB Review

- Most instructions length 16-bit (a few 32-bit)
- Some operands (sp, lr, pc) implicit
  Can't always update sp or pc anymore.
- Only r0-r7 accessible normally
  `add`, `cmp`, `mov` can access high regs
- No prefix/conditional execution
- Only two arguments to opcodes



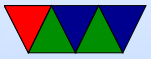
(some exceptions for small constants: `add r0,r1,#1`)

- 8-bit constants rather than 12-bit
- Makes assumptions about "S" setting flags
  (gas doesn't let you superfluously set it, causing problems if you naively move code to THUMB-2)
- Limited addressing modes
- No shift parameter in ALU instructions

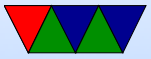


THUMB/ARM interworking

- See `print_string_armthumb.s`
- BX/BLX instruction to switch mode.
  If target is a label, *always* switchmode
  If target is a register, low bit of 1 means THUMB, 0 means ARM
- Can also switch modes with `ldrm`, `ldm`, or `pop` with PC as a destination
  (on armv7 can enter with ALU op with PC destination)

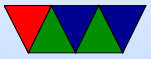


- Can use `.thumb` directive, `.arm` for 32-bit.



THUMB-2

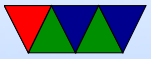
- Extension of THUMB to have both 16-bit and 32-bit instructions
- 32-bit instructions *not* standard 32-bit ARM instructions. It's a new encoding that allows an instruction to be 32-bit if needed.
- All 32-bit ARM instructions have 32-bit THUMB-2 equivalents *except* ones that use conditional execution. The `it` instruction was added to handle this.



- THUMB-2 code can assemble to either ARM-32 or THUMB2

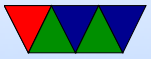
  The assembly language is compatible.

  Common code can be written and output changed at time of assembly.



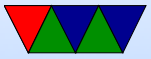
THUMB-2 Coding

- See `test_thumb2.s`
- Use `.syntax unified` at beginning of code
- Use `.arm` or `.thumb` to specify mode

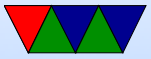


New THUMB-2 Instructions

- BFI – bit field insert
- RBIT – reverse bits
- movw/movh – 16 bit immediate loads
- TB – table branch
- IT (if/then)
- cbz, cbnz – compare and branch if not zero. Only jumps

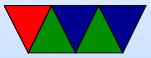


forward



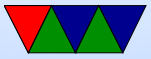
Other THUMB-2 Changes

- Instructions have “wide” and “narrow” encoding.
  Can force this (`add.w` vs `add.n`).
- `rsc` (reverse subtract with carry) removed
- Need to properly indicate “s” (set flags).
  Regular THUMB this is assumed.



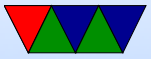
Thumb-2 12-bit immediates

```
top 4 bits 0000 -- 00000000 00000000 00000000 abc
           0001 -- 00000000 abcdefgh 00000000 abc
           0010 -- abcdefgh 00000000 abcdefgh 000
           0011 -- abcdefgh abcdefgh abcdefgh abc
           0100 -- 1bcdedfh 00000000 00000000 000
            ...
           1111 -- 00000000 00000000 00000001 bcd
```



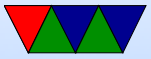
IT (If/Then) Instruction

- Allows limited conditional execution in THUMB-2 mode.
- The directive is optional (and ignored in ARM32)
  the assembler can (in-theory) auto-generate the IT instruction
- ```
  it cc
  addcc %r1,%r2
  ```
- ```
  itete cc
  addcc %r1,%r2
  ```



```
addcs %r1,%r2
addcc %r1,%r2
addcs %r1,%r2
```

- Limit of 4 instructions



Compiler

- `gcc -S hello_world.c`
  On pandarboard creates Thumb-2 by default. Why?

- `gcc -S -march=armv5t -mthumb hello_world.c`
  On my pandaboard, doesn't work. This is because gcc's 16-bit THUMB can't handle the "hard floating point" ABI that is installed on the system.

- `gcc -S -marm hello_world.c`
  On my pandaboard, creates 32-bit ARM code