CS/ECE 5780/6780
Embedded Systems Design

Lecture 2: Architecture, Assembly, and ABI

Thomas Schmid
thomas.schmid@utah.edu

Adapted from Prabal Dutta (prabal@umich.edu)

January 14, 2013
Office Hours

- Anh Luong  Friday 12PM - 1PM  TA Lounge
- Meenakshi B.  Tuesday 10:45AM - 11:45 TA Lounge
- Thomas S.  Tuesday 2PM - 3PM  MEB 3114
Every graduate student has to present on an embedded systems topic

15 minute presentation in the second part of this class

On a topic of your choice

Check out the last two years class pages for examples

Send an email to Thomas Schmid once you have a topic for discussion and approval
Lab Organization

• Groups of 2
  – Make sure you signed up for your group on Canvas
  – We will freeze the groups when you start lab 2
  – You can not switch groups thereafter!

• Try to have mixed groups
  – We have CE, CS, EE, and ME students this year

• Groups not necessarily the same for projects

• Stick to your lab schedule
  – Lab due date is defined by your lab section
  – Check Canvas for your due dates, and observe strict late policy
• There shouldn’t be questions on how to write a piece of code, but more conceptual
• In the first labs, most code is provided. Try to understand it! Don’t just copy-paste it
• Later labs, you will have to write the code!

• Ask **early** if you have questions about certain code in the earlier labs that you don’t understand.
• **Pre-lab:** read through the lab description and answer sheet

• **In-lab:** Detailed, written out, mostly copy paste and procedural.
  – Answer sheet: start filling it out while doing the in-lab

• **Post-lab:** Mostly goes deeper into the In-lab part.
Review HW 1.1

• Decaps, Bypass Capacitor
  – used to filter noise on power rail due to switching
  – place electrically and physically as close to chip as possible
  – see “PCB Design Tutorial” section “Good Bypassing”

• Cleanliness of schematic… it’s Art!
  – *Do not rotate* VCC and GND symbols
  – VCC always up, GND down

• Footprint Confusion
  – Many companies have their own names (e.g. TI)
  – Sometimes hard to find the right one in IPC Wizard

• LED Circuit
  – Resistor defines amount of current through LED
  – Current specifies brightness
The MSP430AFE253 was chosen for its onboard memory and high resolution Analog-to-Digital converters. An important feature is its SPI interface which allows for communication with a BeagleBone Black.

Randy Kellen Madsen
U0412706
ECE 6780-001
LAB SECTION 002

Title: SANDIA CLINIC PROJECT 2013-2014

<table>
<thead>
<tr>
<th>Size</th>
<th>Number</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1/10/2014</td>
<td>002</td>
</tr>
</tbody>
</table>

Date: 1/10/2014
Sheet of
File: C-Users\Sandia SchDoc
Green 1206 LED

Schematic from wikipedia.org

max $I_f = 30 \text{ mA}$  
Luminosity at 10 mA = 6 mcd

Schematic from wikipedia.org

Blue 1206 LED

max \( I_f \) = 20 mA

Luminosity at 20 mA = 30 mcd

Schematic from wikipedia.org

LED Circuit - LiteOn LTST-C150 TBKTK

Luminosity at 10 mA = 6 mcd
Luminosity at 20 mA = 30 mcd

Green

Blue
More HW 1.1

• What is a BOM
  – Bill of Material
  – Used to order parts for your board

• Packages
  – Use TSOP if possible
  – QFN works, but more difficult to solder
  – DO NOT CHOOSE BGA or LGA!

• You won’t use this board for the labs!
Homework 1.2

- Routing
- Making Gerbers
- Due Friday, January 17th, 11:59pm
Low-Power SDR Platform by S. Kuo, University of Michigan
Addressing Modes

- **Offset Addressing**
  - Offset is added or subtracted from base register
  - Result used as effective address for memory access
  - \([<\text{Rn}>, <\text{offset}>]\)

- **Pre-indexed Addressing**
  - Offset is applied to base register
  - Result used as effective address for memory access
  - Result written back into base register
  - \([<\text{Rn}>, <\text{offset}>]\)!

- **Post-indexed Addressing**
  - The address from the base register is used for the memory access
  - The offset is applied to the base and then written back
  - \([<\text{Rn}>], <\text{offset}>\)
<offset> options

• An immediate constant
  - #10

• An index register
  - <Rm>

• A shifted index register
  - <Rm>, LSL #<shift>

• Lots of weird options…
LDR (register)

Encoding T1  All versions of the Thumb instruction set.
LDR<
\text{c}\nRt>,[<Rn>,<Rm>]

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1 1 0 0</td>
</tr>
</tbody>
</table>

\(t = \text{UInt}(Rt); \ n = \text{UInt}(Rn); \ m = \text{UInt}(Rm);
\)
\(\text{index} = \text{TRUE}; \ \text{add} = \text{TRUE}; \ \text{wback} = \text{FALSE};
\)
\((\text{shift}_t, \ \text{shift}_n) = (\text{SRT}ype_{\text{LSL}}, 0)\);

Encoding T2  ARMv7-M
LDR<\text{c}\.W \text{Rt}>,[<Rn>,<Rm>{,LSL \#<imm2>}]}

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

if \(Rn = \text{'1111'}\) then SEE LDR (literal);
\(t = \text{UInt}(Rt); \ n = \text{UInt}(Rn); \ m = \text{UInt}(Rm);
\)
\(\text{index} = \text{TRUE}; \ \text{add} = \text{TRUE}; \ \text{wback} = \text{FALSE};
\)
\((\text{shift}_t, \ \text{shift}_n) = (\text{SRT}ype_{\text{LSL}}, \text{UInt}(\text{imm2}))\);
if \(m \text{ IN } \{13,15\}\) then UNPREDICTABLE;
if \(t = 15 \&\& \text{InITBlock()} \&\& !\text{LastInITBlock()}\) then UNPREDICTABLE;
Assembler syntax

LDR<
<
<
[, LSL #<shift>]

where:

<
> See Standard assembler syntax fields on page A7-207.

<
> Specifies the destination register. This register is allowed to be the SP. It is also allowed to be the PC, provided the instruction is either outside an IT block or the last instruction of an IT block. If it is the PC, it causes a branch to the address (data) loaded into the PC.

<
> Specifies the register that contains the base value. This register is allowed to be the SP.

<
> Contains the offset that is shifted left and added to the value of <
> to form the address.

<
> Specifies the number of bits the value from <
> is shifted left, in the range 0-3. If this option is omitted, a shift by 0 is assumed and both encodings are permitted. If this option is specified, only encoding T2 is permitted.
LDR (immediate)

Encoding T1  All versions of the Thumb instruction set.
LDR<
<Rt>, [<Rn>{,#<imm5>}]}

| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------------------|---------|
| 0 1 1 | imm5 | Rn | Rt |

t = UInt(Rt);  n = UInt(Rn);  imm32 = ZeroExtend(imm5:'00', 32);
index = TRUE;  add = TRUE;  wback = FALSE;

Encoding T2  All versions of the Thumb instruction set.
LDR<, [SP{,#<imm8>]}]

| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------------------|---------|
| 1 0 0 1 | Rt | imm8 |

t = UInt(Rt);  n = 13;  imm32 = ZeroExtend(imm8:'00', 32);
index = TRUE;  add = TRUE;  wback = FALSE;

Encoding T3  ARMv7-M
LDR<.W <Rt>, [<Rn>{,#<imm12>}]}

| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------------------|---------|
| 1 1 1 1 | 0 0 | 1 1 0 1 | Rn | Rt | imm12 |

if Rn == '1111' then SEE LDR (literal);
t = UInt(Rt);  n = UInt(Rn);  imm32 = ZeroExtend(imm12, 32);
index = TRUE;  add = TRUE;  wback = FALSE;
if t == 15 && InITBlock() && !LastInITBlock() then UNPREDICTABLE;

Encoding T4  ARMv7-M
LDR< <Rt>, [<Rn>,#<imm8>]
LDR< <Rt>, [<Rn>],#<imm8>
LDR< <Rt>, [<Rn>,#<imm8>]!
start:
movs r0, #1
movs r1, #0
movs r2, #1
mul s r0, r0, r1
beq done
movs r2, #2
done:
b done
Questions?

Comments?

Discussion?

- One point that really **sticks out** about this lecture
- One point that was **unclear** about this lecture