Lab practical 1 review

Next week

Tim Rogers 2022

Lab Schedule this Week

- This week: Practical Exams in Lab
    - You may go to Open Lab / Office Hours at regular times
    - You should go to your normal lab section.
    - Other than that, please stay out.
        - No personal access to lab outside Office Hours
        - i.e., you may not enter lab from 5:30pm to 7pm or 10pm to 11:30am.
        - If you are an ECE 270 UTA, do not use the side door.
        - Why? Preparation and exceptions for students with accommodations.

No Talking about the Practical

- After Monday, you will not talk to anyone about the lab practical exam until I tell you grades.
  - Not all students will take the practical this week
  - Consequences are severe

Practical 1 Format

No access to: course notes or the internet

- Part 1: Concept questions (20 - 30 of them)
  - These will be similar in style to prelab/homework/quiz questions.
  - Mostly free response. Some will be multiple-choice with dozens of options.
  - You get softcopy versions of the five reference manuals.
  - This part has a weight of 10 (about 10% of the course grade)
- Part 2: Program the development board
  - We supply the board and do all wiring in advance.
  - We give you main.s file with instructions and test code and an unhelpful autotest.o file
  - This part has a weight of 12 (about 12% of the course grade)
  - You will have a total of two hours to work.
  - Do either part first, go back and forth, etc.
  - When time is up, we will collect your saved results.

You may bring...

- Nothing but your student ID.
  - No calculator.
  - No phone.
  - No web browser.
  - No notes.
  - No pencil.
  - No pen.
- You will place your belongings at the front of the room by the window before you go to your assigned lab station.
  - We will give you a pencil and a sheet of blank paper in trade for your ID. Have it out and ready.

When you arrive

- Go to your assigned station
- Read the instructions on the screen
- Do not type anything until your TA says ‘start

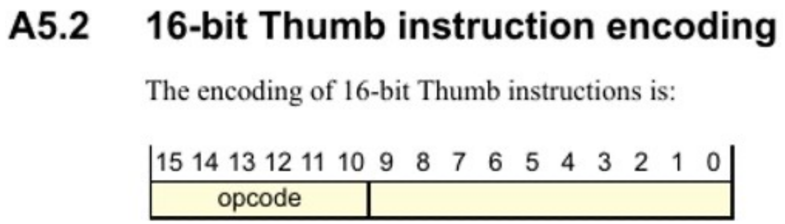
Preparation

- Redo your quizzes, labs, and homeworks
- Try using the calculator on the lab workstations

Instruction encoding/decoding

- Do you remember how to decode a 16-bit value to find the instruction?
- Do you remember how to take an instruction and encode it into a 16-bit value?

Disassembly Example

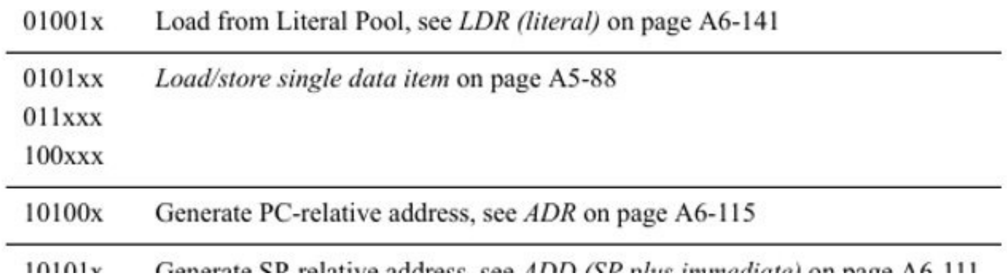


A5.2 16-bit Thumb instruction encoding

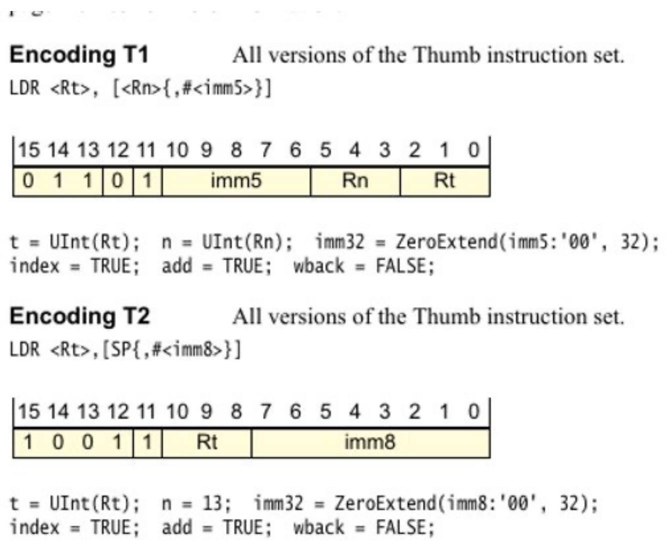
The encoding of 16-bit Thumb instructions is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opcode															

- What instruction is 9f19?
- 1001 1111 0001 1001
- 100111 11 0001 1001
- 1001 111 100011001
- 1001 1 111 00011001



01001x	Load from Literal Pool, see *LDR (literal)* on page A6-141
0101xx 011xxx 100xxx	*Load/store single data item* on page A5-88
10100x	Generate PC-relative address, see *ADR* on page A6-115
10101x	Generate SP-relative address, see *ADD (SP plus immediate)* on page A6-111



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	0	1	imm5					Rn			Rt		

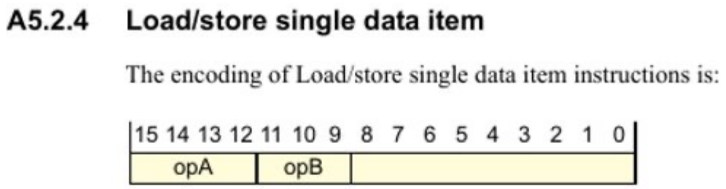
```
t = UInt(Rt);  n = UInt(Rn);  imm32 = ZeroExtend(imm5:'00', 32);
index = TRUE;  add = TRUE;  wback = FALSE;
```

Encoding T2 All versions of the Thumb instruction set.

```
LDR <Rt>,[SP{,#<imm8>}]
```

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

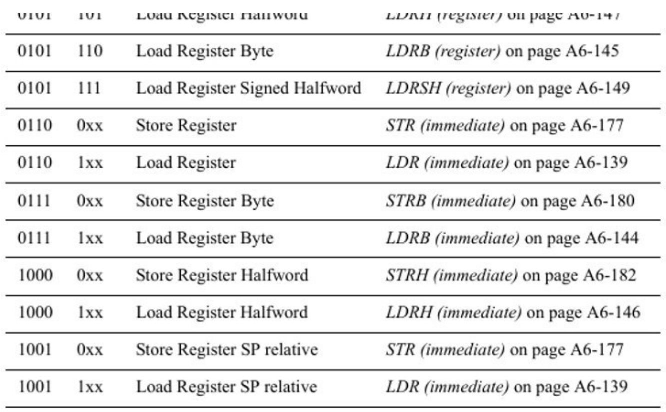
```
t = UInt(Rt);  n = 13;  imm32 = ZeroExtend(imm8:'00', 32);
index = TRUE;  add = TRUE;  wback = FALSE;
```



A5.2.4 Load/store single data item

The encoding of Load/store single data item instructions is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opA				opB											

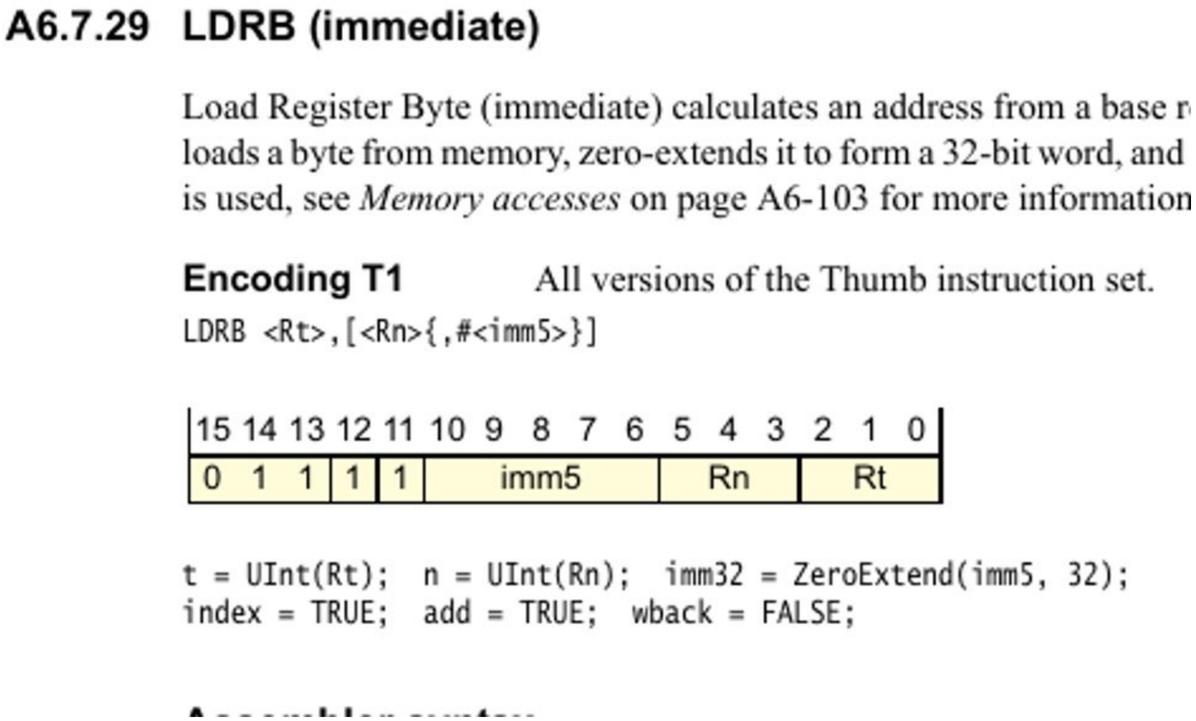


0101	101	Load Register Halfword	*LDRH (register)* on page A6-147
0101	110	Load Register Byte	*LDRB (register)* on page A6-145
0101	111	Load Register Signed Halfword	*LDRSH (register)* on page A6-149
0110	0xx	Store Register	*STR (immediate)* on page A6-177
0110	1xx	Load Register	*LDR (immediate)* on page A6-139
0111	0xx	Store Register Byte	*STRB (immediate)* on page A6-180
0111	1xx	Load Register Byte	*LDRB (immediate)* on page A6-144
1000	0xx	Store Register Halfword	*STRH (immediate)* on page A6-182
1000	1xx	Load Register Halfword	*LDRH (immediate)* on page A6-146
1001	0xx	Store Register SP relative	*STR (immediate)* on page A6-177
1001	1xx	Load Register SP relative	*LDR (immediate)* on page A6-139

- LDR(SP) Rt   #offset   =>   LDR R7, [SP, #100]

Assembly Example

- Assemble the following instruction:
  - LDRB R7,[R6,#8]
  - Rt = 7 (111)
  - Rn = 6 (110)
  - Imm5 = 8 (01000)
- 011 1 1 01000 110 111
- 0111 1010 0011 0111 => 7a37



A6.7.29 LDRB (immediate)

Load Register Byte (immediate) calculates an address from a base r
loads a byte from memory, zero-extends it to form a 32-bit word, and
is used, see *Memory accesses* on page A6-103 for more information

Encoding T1 All versions of the Thumb instruction set.

```
LDRB <Rt>,[<Rn>{,#<imm5>}]
```

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

```
t = UInt(Rt);  n = UInt(Rn);  imm32 = ZeroExtend(imm5, 32);
index = TRUE;  add = TRUE;  wback = FALSE;
```

Assembler syntax

Machine instructions

- Move instruction
- Arithmetic instructions
- Logical instructions
- Shift/rotate instructions
- Control flow instructions (B, B**, BL, BX)
- Load & Store instructions
  - Load literal
- Stack instructions (PUSH, POP, ADD, SUB)
- Why do some instructions have an ‘S’ suffix?
  - Because they Set the flags. (Except for the CMP instruction which has no suffix.)

Addressing Modes

- Register values
- Immediate values
- Some instructions support only one form.
  - e.g., ORRS R0,R1
- Some support both.
  - e.g., MOVS R0,R1; MOVS R1,#4

Assembler Directives

- .cpu
- .thumb
- .syntax
- .text
- .data
- .global symbol
- .word value
- .space size
- .string "…"
- .balign boundary
- .equ name, replacement
- .thumb_set
- .type <label>, %function

Assembler Labels

- A label is a symbolic name for an address.
- An EQU is a symbolic name for anything.
  - Similar to a #define in C

- Neither one of these things causes space to be reserved in memory.
  - A label is like a bookmark for a memory location.
  - An EQU is a symbolic substitution.

Translation from C to assembly

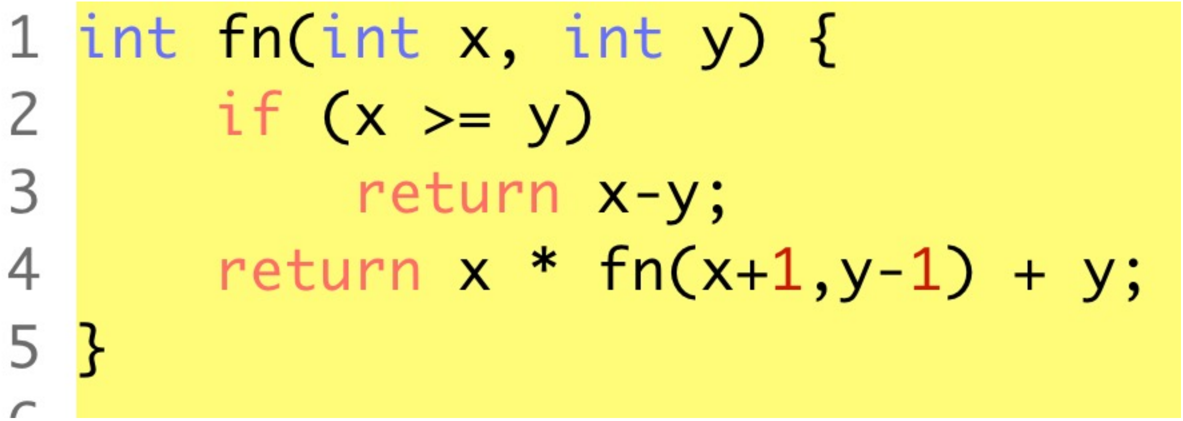
- Simple statements
- Representation of variables as registers
- Representation of variables in the data segment
  - How do you load and store a value to a word of memory in the data segment whose address is represented by a label?
- if-then-else
- do-while loops
- while loops
- for loops

Using the Stack

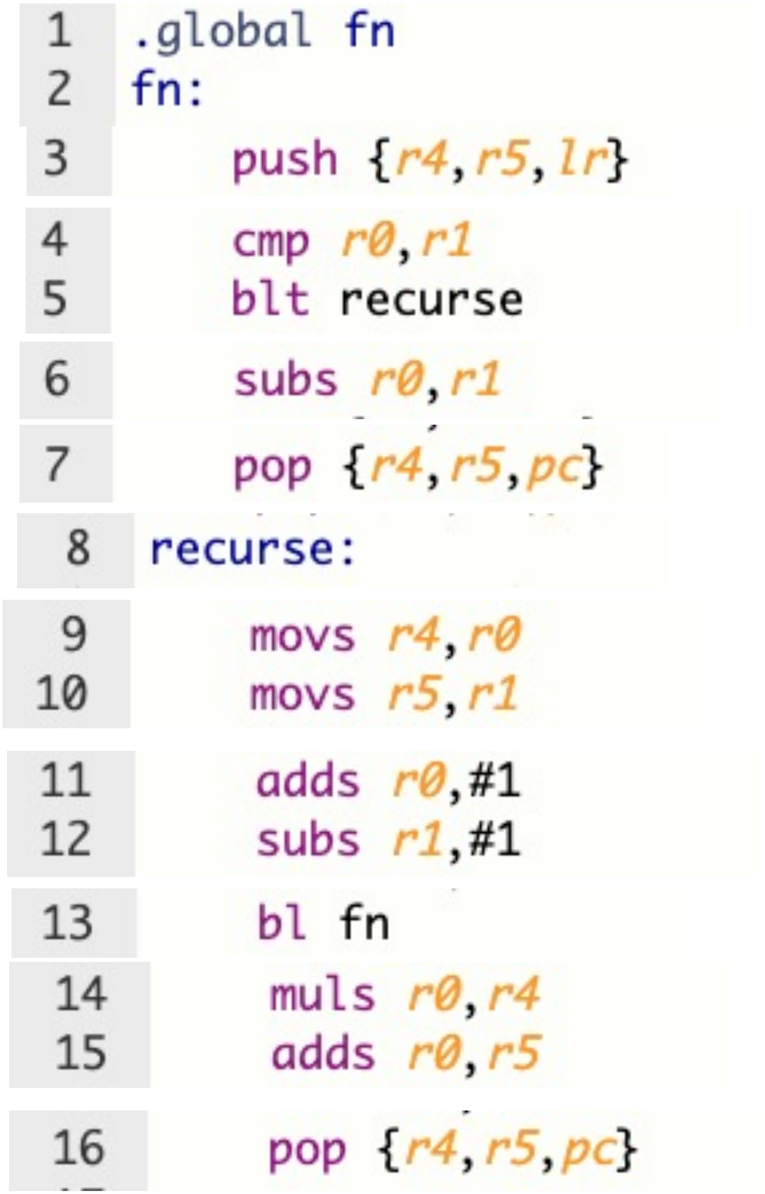
- How do subroutines work?
  - Conforming to the Application Binary Interface.
  - How do you pass parameters to a subroutine?
  - How do you return a value from a subroutine?
    - Even if a subroutine does not return a value, it must still return.
- How do push and pop work?
  - When and why do we need to use them?
- Growing and shrinking the stack
  - Where does it start?
  - Which way does it grow?

Recursive Subroutines

- Would it be a problem to translate this function to assembly?



```c
int fn(int x, int y) {
    if (x >= y)
        return x-y;
    return x * fn(x+1,y-1) + y;
}
```



```
.global fn
fn:
    push {r4,r5,lr}
    cmp r0,r1
    blt recurse
    subs r0,r1
    pop {r4,r5,pc}
recurse:
    movs r4,r0
    movs r5,r1
    adds r0,#1
    subs r1,#1
    bl fn
    muls r0,r4
    adds r0,r5
    pop {r4,r5,pc}
```

Memory Reference Nuances

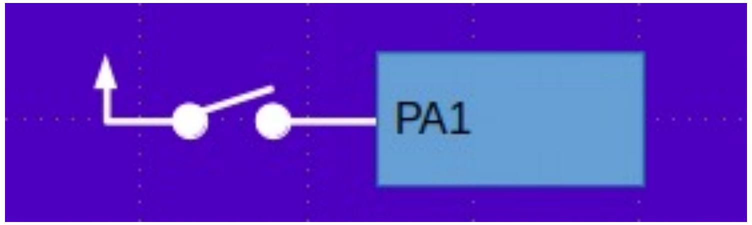
- Alignment
    - What does it mean? Where is it needed?
- Endianness
    - What does it mean? When does it matter?

General Purpose I/O

- Be familiar with the memory map and layout of control registers.
- What is the RCC?
  - What does it do?
  - How do you configure it?
- What are some of the GPIO control registers you will use and why?
  - How do you configure a pin to be an input?
  - How do you configure a pin to be an output?
  - How do you set a specific output pin high?
  - How do you set a specific output pin low?
  - How do you pull input pins high or low?
  - How do you set the output type? The output speed?

Pull-up / Pull-down resistors

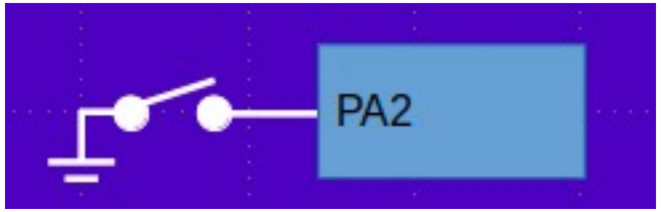
- Gently pull the input high or low in cases where it is not being driven. E.g.:



- Maybe we want PA1 to read ‘low’ unless the switch is pressed.
- Configure the PUPDR to pull it low.

Pull-up / Pull-down resistors

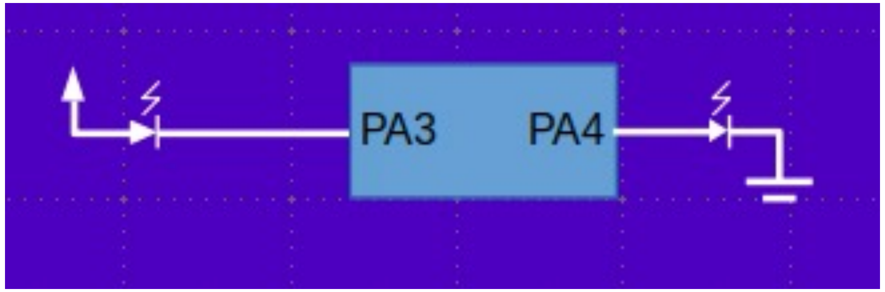
- Gently pull the input high or low in cases where it is not being driven. E.g.:



- Maybe we want PA2 to read ‘high’ unless the switch is pressed.
- Configure the PUPDR to pull it high.

Driving LEDs

- What logic values on PA3 and PA4 will illuminate the LEDs?



Which pins do what?

- Which pin can be configured for DAC_OUT1?
  - You can look this up, right?
    - Device Datasheet, Table 13, pp 31–36, shows the pin functions.
- The DAC, itself, won't be covered on the practical, but it is reasonable to be asked about the address or offset of any of its I/O registers.
  - Device Datasheet, Table 16, pp 40–41, shows base addr.
  - FRM, Table 51, page 293, shows I/O reg offsets for DAC.

Interrupts

- What is an interrupt?
  - It’s a hardware-invoked subroutine.
- What is the one thing an ISR should always do?
  - Acknowledge the interrupt by checking and clearing the specific interrupt flag.
  - What happens if it doesn’t?
- What are the steps to enable an interrupt?
- What registers are saved before invoking an ISR?
- What happens if, while an ISR is running:
  - A higher priority interrupt occurs?
  - A lower priority interrupt occurs?

How does SysTick work?

- What do the RVR, CVR, and CSR registers do?
  - How soon will an interrupt occur if
    - RVR = x
    - CVR = y
    - CSR = z
- Do you need to enable interrupts for SysTick?

How do Timers Work?

- What is a prescaler? An auto-reload register?
  - Should you add one or subtract one when setting these registers?
- What is the upper limit of a timer counter?
- How large (in bits) are PSC,ARR,CNT?
- How do you enable the counter?
- What does the DIER do?
- Why is it necessary to clear the UIF bit of TIMx_SR?
  - And how do you do so?
- If you must configure an update event to occur at a rate of exactly 0.8 Hz, could you do that?
- How about a rate as close as possible to 0.73846153846 Hz?
- How about a rate as close as possible to 0.72917122398681658427 Hz?

How to use Timers

- What steps are necessary to configure a timer to generate an update event interrupt?
- First, enable the RCC clock to the timer.
- Write divisor-1 to TIMx_PSC
- Write count-1 to TIMx_ARR
- OR CEN (0x01) into TIMx_CR1 to enable counter.
- OR UIE (0x01) into TIMx_DIER to enable interrupt.
- Write 1<<TIMx_IRQn to NVIC_ISER to unmask interrupt.

Final advice

- Do your previous homework and pre labs (for concepts)
- Practice programming your labs – without looking at your solutions
  - Just to remind yourself what you did and prove you can do it again, faster
- Don’t panic ☺