# Microcontroller Architecture

- core is ARM Cortex M0+ (32 bits core)
- Peripherals for embedded interfacing and control
  - Analog IO
  - Digita IO
  - Timing
  - Clock generators
  - Communications

Thumb2 instruction set
- slides
- ARM architecture V6-M reference Manual

## Core Architecture

Load / store

The only memory operations are load and store

Data processing only through registers

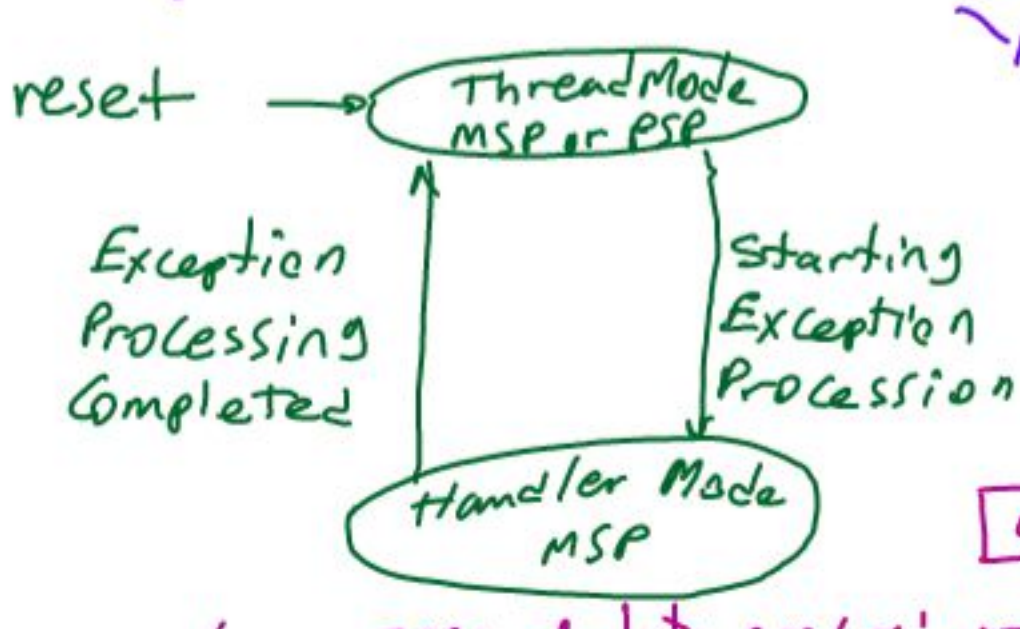
13 General purpose registers

R0 — R12 (mainly R0 — R7 are used)

plus SP (R13), LR (R14), PC (R15)

plus status and control registers

→ two stack Pointers
- MSP main
- PSP process




example Instruction format

Add/Sub register

| 7 | 3 | 3 | 3 |
|---|---|---|---|
| opcode | Rm | Rn | Rd |
| | S2 | S1 | dest |

| 9 | | |
|---|---|---|
| opcod | Rm | Rdn |

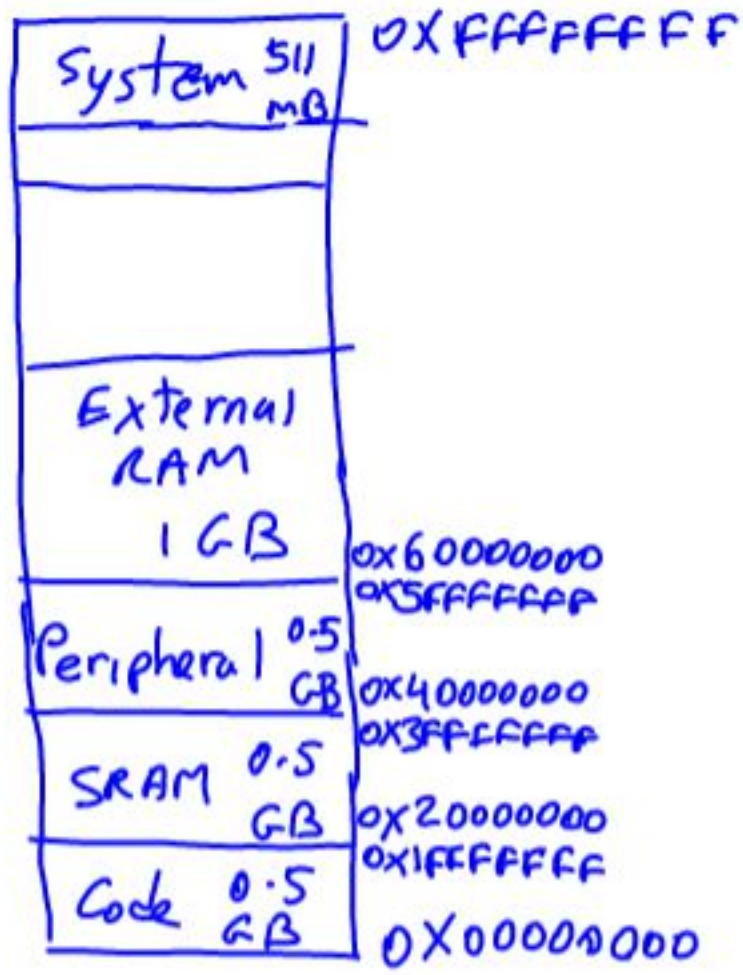
ADD, Comp, MOV

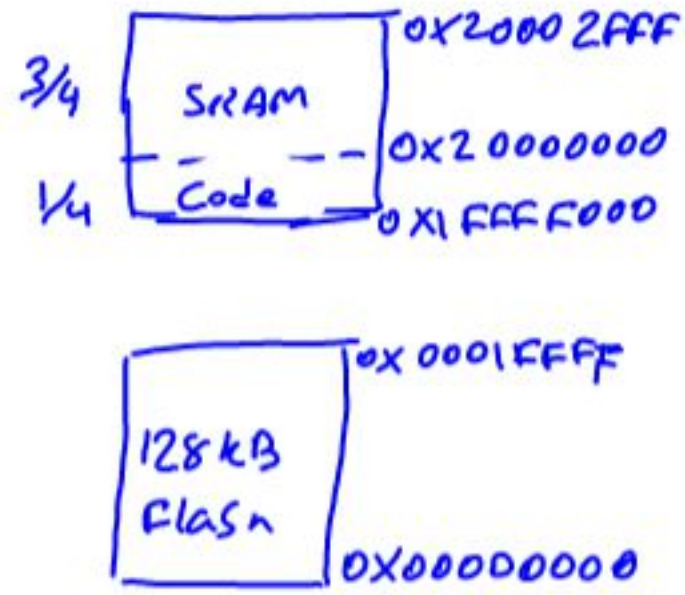
special data processing

Memory Map → Slide 11

4 Gbytes of Memory Space

Memory Mapped IO




Little Endian

Thumb Instructions

ARM instruction set 32 bit instructions

Thumb is mostly a 16 bit version of that

very few 32 bit instructions

examples:

ADD <Rd>, <Rn>, <Rm>

<Rd> = <Rn> + <Rm>

ADDS <Rd>, <Rn>, <Rm>

affects the flags

flags: N, Z, C, V

ADD <Rd>, <Rm>  may use all registers for Rd

<Rd> = <Rd> + <Rm>

# Instruction Set Summary  Slide 19

a nice overview may be found in "The definitive guide to the ARM Cortex-M0" by: Joseph Yiu chapters 5 and 6

Load and Store

memory addressing

Offset

indexed

LDR <Rt>, <Rn>, <Rm>

<Rt> = Mem [<Rn> + <Rm>]

STR <Rt>, <Rn>, # immed

Mem[<Rn> + immed] = <Rt>

Sign extending possible

MOV reg to reg

MOV immed to reg

Load Literal value into register

refer to "Thum 2 Assembly" pdf file

LDR <rd>, = value

Decimal: 3909

Hexadecimal: 0xa7ee

Character: 'A'

String: "44??"

Notes from "Thumb2 Assembly"

## MOVING DATA

high → MOV Rd, Rm
MOV Rd, #immed8

## MEM ACCESS

indexed addressing

LDR Rt, [Rn, Rm]
LDRH
LDRB

STR Rt, [Rn, Rm]
STRH
STRB

Offset addressing

LDR Rt, [Rn, #immed5]  STR Rt, [Rn, #immed5]

## STACK ACCESS

PUSH {Ra, Rb, ...}
PUSH {R1-R4, LR}

mem [SP-4] = Ra
mem [SP-8] = Rb
⋮

POP { }
POP

Ra = mem [SP]
Rb = mem[SP+4]

## ARITHMETIC

ADD Rd, Rn, Rm
ADD Rd, Rn, #immed3
ADD Rd, #immed8
high → ADD Rd, Rm
SUB, NEG, MUL, CMP

## LOGICAL

AND Rd, Rm
ORR, EOR, BIC, MVN

Rd = AND(Rd, NOT(Rm))
Rd = NOT (Rm)

## SHIFT

LSR  LSL Rd, Rm  Rd = Rd << Rm
LSR  LSL Rd, Rm, #immed5  Rd = Rm << immed5

ASR Rd, Rm
ASR Rd, Rm, #immed5

SIGN EXTEND

SXTB Rd, Rm  Rd = SignExtend(Rm[7:0])
SXTH Rd, Rm

## Program Flow Control

B <label>  ± 2046 bytes relative
BX Rm

B <cond><label>  ± 254 relative

BEQ TEST
=
TEST =
=

BL <label>  ± 16MB relativ (32 bits instr)
BLX Rm

=
BL func1
=
=

func1: =
=
=
BX LR

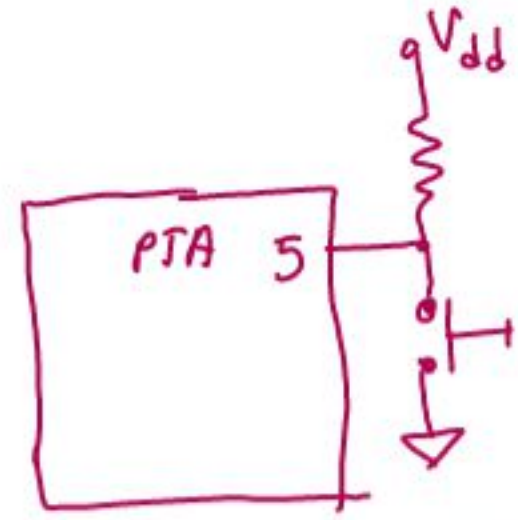
## PSEUDO INSTRUCTION

LDR Rd, =immed32
LDR Rd, label

label and literal

```
LDR  R3, =MY_NUMBER   ; Get the memory location of MY_NUM
LDR  R4, [R3]         ; Get the value 0X12345678
...
LDR  R0, =HELLO_TEXT  ; Get the starting address of HELLO_TEXT
BL   PrintText
...
ALIGN 4
MY_NUMBER   DCD  0X12345678
HELLO_TEXT  DCB  "Helo\n", 0 ; Null-terminated string
```

example: Check if switch is pressed
make pin 5 of port A input




0000 0000 0000 0000 0000 0000 0010 0000
n = 0X00000020

PTA_Base 0X400FF000

| | | |
|---|---|---|
| PDOR | offset | 0X0 |
| PSOR | " | 0X4 |
| PCOR | " | 0X8 |
| PTOR | " | 0XC |
| PDIR | " | 0X10 |
| PDDR | " | 0X14 |

Section 41.2
Memory Map and register definition
"KL25 Sub-Family Reference Manual"
KL25P80M48SFORM

---

1 →

```
        LDR  R1, =0X400FF000
        LDR  R2, #0X1
        LSL  R2, R2, #0X5
        LDR  R3, 0X14[R1]
        BIC  R3, R2
        STR  R3, 0X14[R1]   ← make bit 5 input
CHECK:  LDR  R4, 0X10[R1]
        AND  R4, R2
        BEQ  PRESSED
        BNE  NOTPRESSED
        BR   CHECK
```

absolute addressing

1

```
LDR  R1, =0X40008038
LDR  R2, #0[R1]
LDR  R3, #1
LSL  R3, R3, #9
ORR  R2, R2, R3
STR  R2, #0[R1]
```

clock
Section 12.2
SIM_SCGC5 at address 4000 8038