

# TI-RSLK

Texas Instruments Robotics System Learning Kit



TEXAS INSTRUMENTS



# Module 3

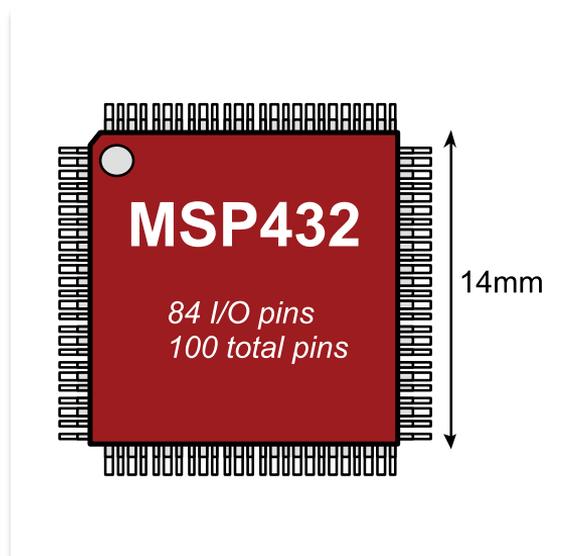
Lecture: ARM Cortex M - Architecture



# ARM Cortex M Architecture

## You will learn in this module

- Cortex M Architecture
  - Buses
  - CISC versus RISC
  - Registers
  - Memory
  - Addressing modes



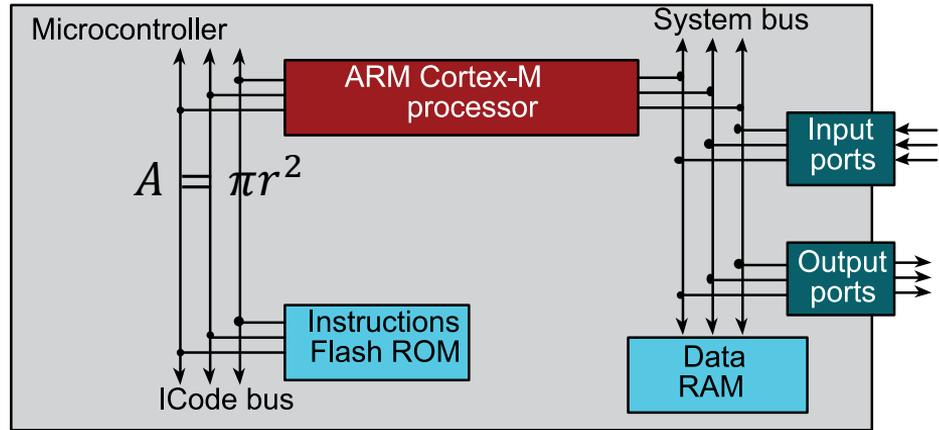


# ARM Cortex M Architecture

## ARM Cortex-M4 processor

- Harvard versus von Neumann architecture
- Different busses for instructions and data

- ICode bus - Fetch op codes from ROM
- System bus - Data from RAM and I/O
- Dcode bus - Debugging
- PPB bus - Private peripherals





# Reduced Instruction Set Computer (RISC)

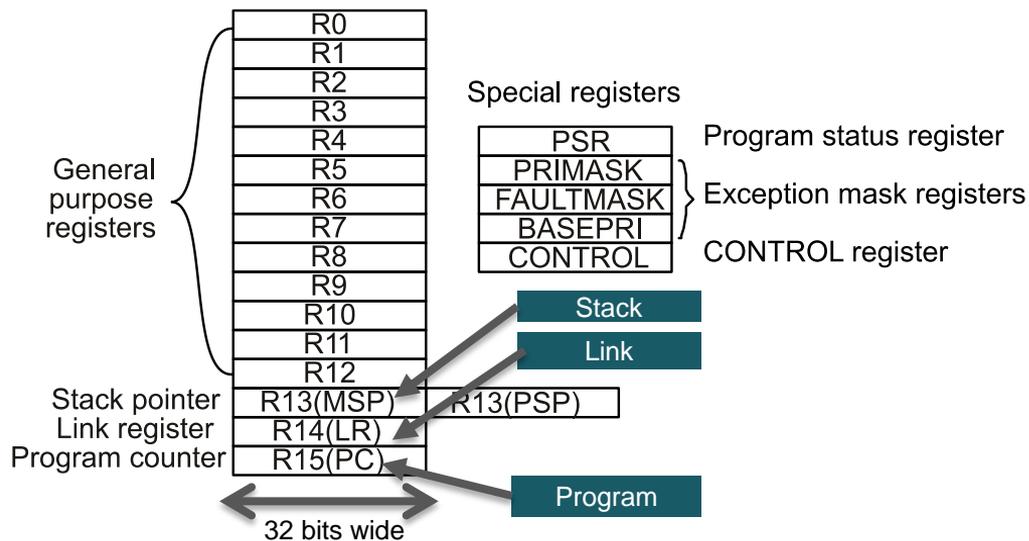
CISC	RISC
Many instructions	Few instructions
Instructions have varying lengths	Instructions have fixed lengths
Instructions execute in varying times	Instructions execute in 1 or 2 bus cycles
Many instructions can access memory	Few instructions can access memory <ul style="list-style-type: none"><li>• Load from memory to a register</li><li>• Store from register to memory</li></ul>
In one instruction, the processor can both <ul style="list-style-type: none"><li>• Read memory and</li><li>• Write memory</li></ul>	<ul style="list-style-type: none"><li>• No one instruction can both read and write memory in the same instruction</li></ul>
Fewer and more specialized registers <ul style="list-style-type: none"><li>• Some registers contain data</li><li>• Others contain addresses</li></ul>	<ul style="list-style-type: none"><li>• Many identical general purpose registers</li></ul>
Many different types of addressing modes	Limited number of addressing modes <ul style="list-style-type: none"><li>• Register, PC - relative</li><li>• Immediate</li><li>• Indexed</li></ul>

## RISC machine

- Pipelining provides single cycle operation for many instructions
- Thumb-2 configuration employs both 16 and 32-bit instructions



# Registers



## Where are data?

- Registers
- RAM
- ROM
- I/O ports

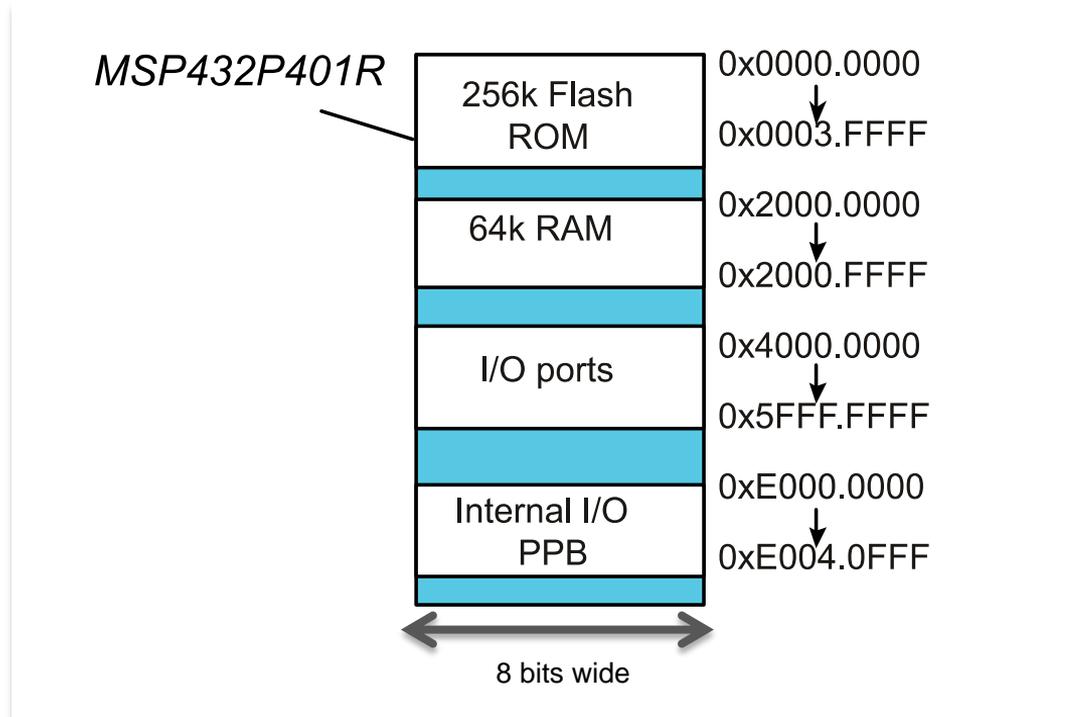
## Where are commands?

- ROM (pointed to by PC)

<i>Condition Code Bits</i>	<i>Indicates</i>
N negative	Result is negative
Z zero	Result is zero
V overflow	Signed overflow
C carry	Unsigned overflow



# Memory Map



For the detailed Memory Map go to <http://www.ti.com/lit/ds/symlink/msp432p401r.pdf>



# Endianness

16-bit 1000 = 0x03E8

Address	Data
0x2000.0850	0x03
0x2000.0851	0xE8

Big Endian

Address	Data
0x2000.0850	0xE8
0x2000.0851	0x03

Little Endian

32-bit 1000000 = 0x000F4240

Address	Data
0x2000.0850	0x00
0x2000.0851	0x0F
0x2000.0852	0x42
0x2000.0853	0x40

Big Endian

Address	Data
0x2000.0850	0x40
0x2000.0851	0x42
0x2000.0852	0x0F
0x2000.0853	0x00

Little Endian

ASCII string "Jon" = 0x4A,0x6F,0x6E,0x00

Address	Data
0x2000.0850	0x4A
0x2000.0851	0x6F
0x2000.0852	0x6E
0x2000.0853	0x00

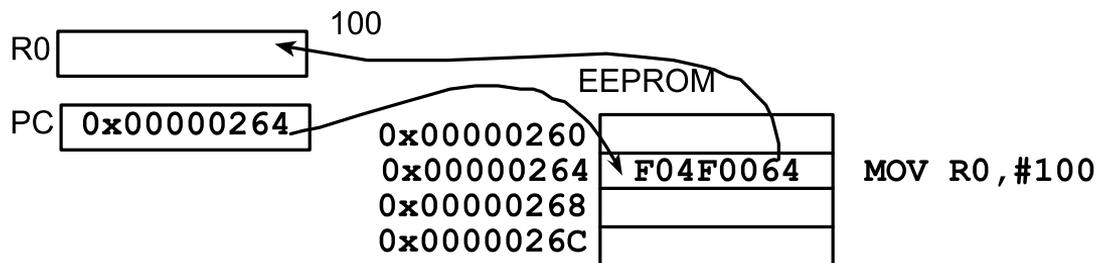
Big Endian and Little Endian



# Addressing Modes: immediate

Register  
Immediate  
Indexed  
PC-relative

**MOV R0, #100**

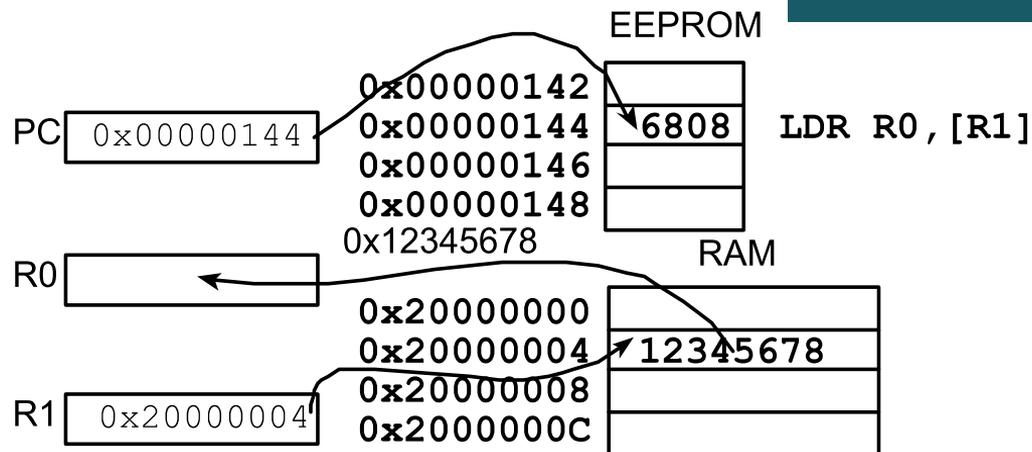




# Addressing Modes: Indexed

```
LDR R0, [R1]
```

Register  
Immediate  
Indexed  
PC-relative

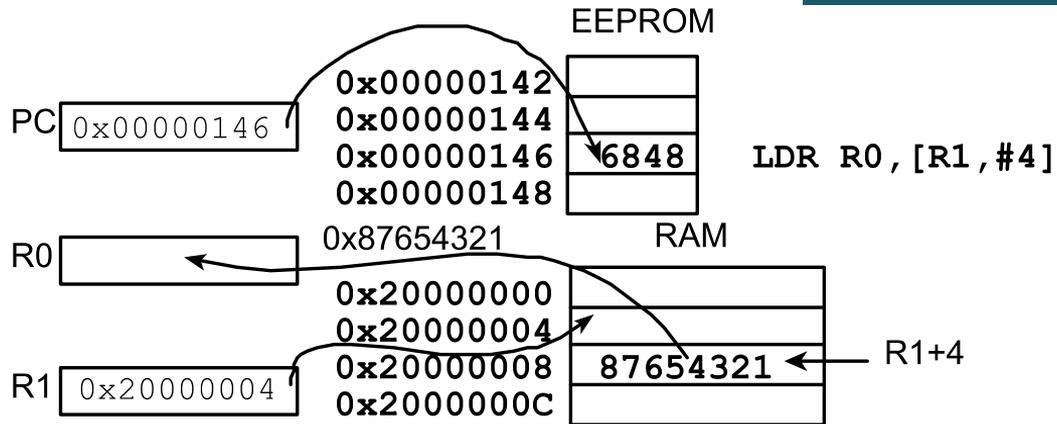




# Addressing Modes: Indexed

```
LDR R0, [R1, #4]
```

Register  
Immediate  
Indexed  
PC-relative





# Variable Access: Load/store architecture

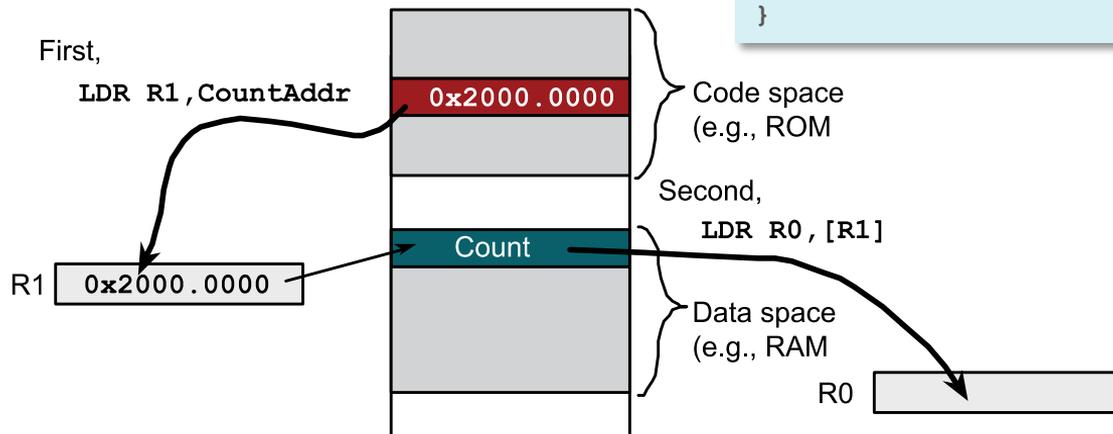
```
Increment: .asmfunc
    LDR R1,CountAddr ; R1=pointer to Count
    LDR R0,[R1]      ; R0=value of Count
    ADD R0,R0,#1
    STR R0,[R1]
    BX LR
.endasmfunc
CountAddr .field Count,32
```

[PC,#8]

0x20000000

Register  
Immediate  
Indexed  
PC-relative

```
uint32_t Count;
void Increment(void) {
    Count++;
}
```





# ARM Cortex M Architecture

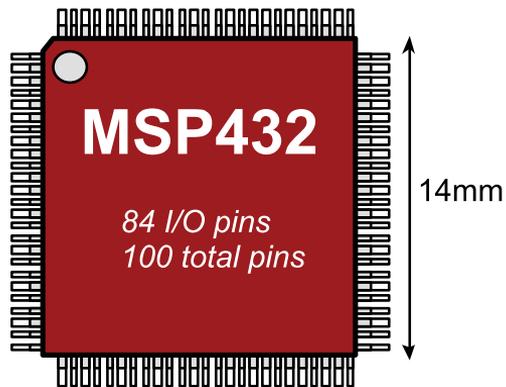
## Summary

- Architecture
  - Buses
  - Registers
  - Memory
  - Addressing modes

Register  
Immediate  
Indexed  
PC-relative

## Terms:

- RISC vs CISC
- Little vs big endian
- Address vs data
- Variables





# Module 3

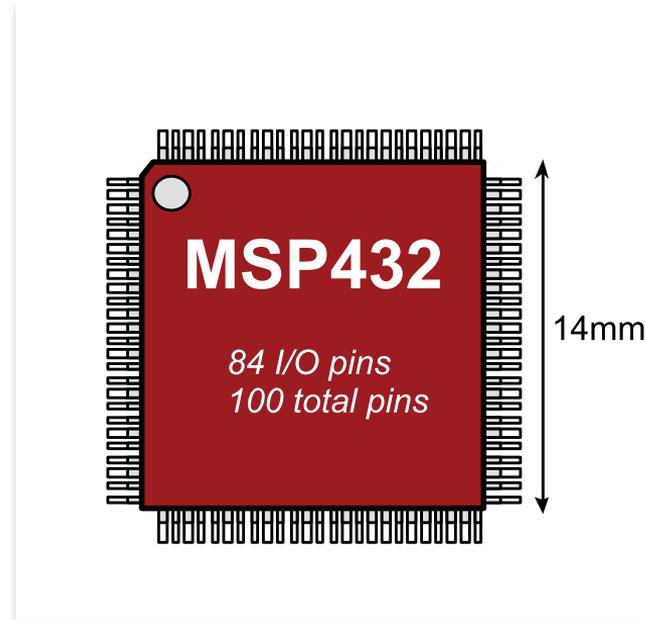
Lecture: ARM Cortex M Assembly Programming



# ARM Cortex M Assembly Programming

## You will learn in this module

- Assembly Programming
  - Logical and shift operations
  - Addition, subtraction, multiplication and divide
  - Accessing memory
  - Stack
  - Functions, parameters
  - Conditionals
  - Loops





# Logic Operations

A Rn	B Operand2	A&B AND	A B ORR	A^B EOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

AND {Rd, } Rn, <op2> ;Rd=Rn&op2  
 ORR {Rd, } Rn, <op2> ;Rd=Rn|op2  
 EOR {Rd, } Rn, <op2> ;Rd=Rn^op2

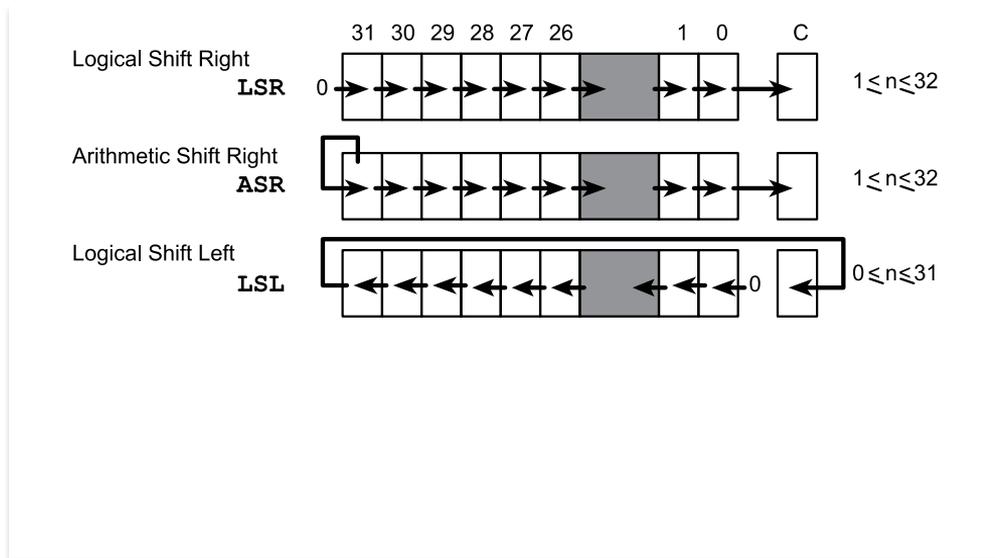
<op2>

- Register
- Register, shifted
- Constant

ORR R0,R1,R2								
R1	0001	0010	0011	0100	0101	0110	0111	1000
R2	1000	0111	0110	0101	0100	0011	0010	0001
ORR	1001	0111	0111	0101	0101	0111	0111	1001



# Shift Operations



```

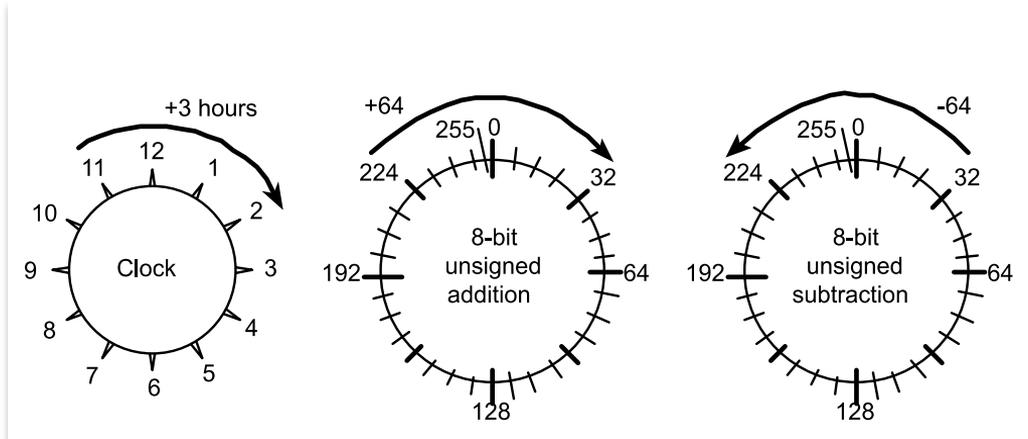
LSR Rd, Rm, Rs ; logical shift right Rd=Rm>>Rs      (unsigned)
LSR Rd, Rm, #n ; logical shift right Rd=Rm>>n      (unsigned)
ASR Rd, Rm, Rs ; arithmetic shift right Rd=Rm>>Rs  (signed)
ASR Rd, Rm, #n ; arithmetic shift right Rd=Rm>>n  (signed)
LSL Rd, Rm, Rs ; shift left Rd=Rm<<Rs              (signed, unsigned)
LSL Rd, Rm, #n ; shift left Rd=Rm<<n              (signed, unsigned)

```



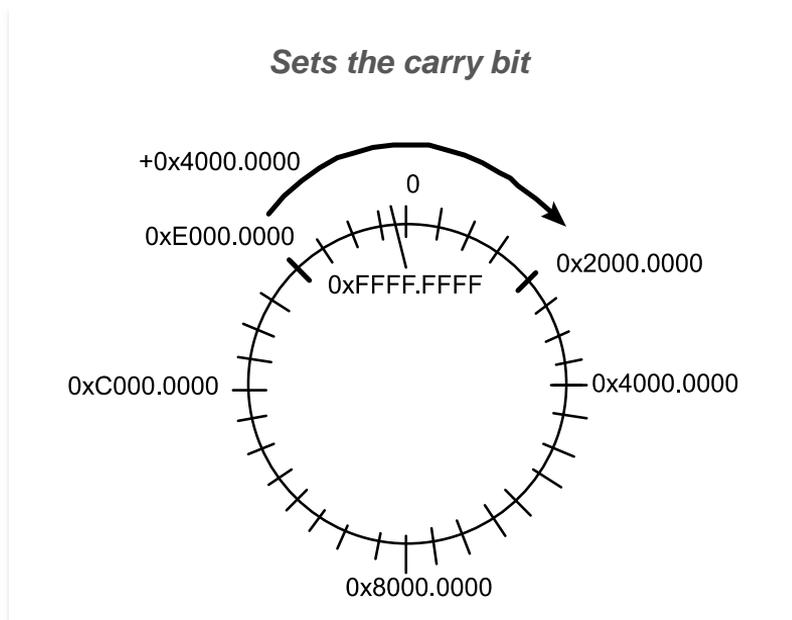
# Arithmetic Operations

- Addition/subtraction
  - Two n-bit  $\rightarrow$  n+1 bits
- Multiplication
  - Two n-bit  $\rightarrow$  2n bits
- Avoid overflow
  - Restrict input values
  - Promote to higher, perform, check, demote
- Division
  - Avoid divide by 0
  - Watch for dropout
- Signed versus unsigned
  - Either signed or unsigned, not both
  - Be careful about converting types





# Addition and Subtraction



## Condition

### Code Bits

### Indicates

N	negative	Result is negative
Z	zero	Result is zero
V	overflow	Signed overflow
C	carry	Unsigned overflow

## <op2>

- Register
- Register, shifted
- Constant

```

ADD  {Rd, }  Rn,  <op2>  ;Rd = Rn + op2
SUB  {Rd, }  Rn,  <op2>  ;Rd = Rn - op2
CMP  Rn,    <op2>  ;Rn - op2
  
```



# Multiplication and Division

```
MUL   {Rd,}  Rn,  Rm      ;Rd = Rn * Rm
UDIV  {Rd,}  Rn,  Rm      ;Rd = Rn/Rm unsigned
SDIV  {Rd,}  Rn,  Rm      ;Rd = Rn/Rm signed
```

```
uint32_t N,M;
// times 0.6
void Fun(void){
    M = 3*N/5;
}
```

```
.data
.align 2
N .space 4
M .space 4
.text
.align 2
Fun: .asmfunc
    LDR R3, NAddr ; R3 = &N (R3 points to N)
    LDR R1, [R3] ; R1 = N
    MOV R0, #3 ; R0 = 3
    MUL R1, R0, R1 ; R1 = 3*N
    MOV R0, #5 ; R0 = 5
    UDIV R0, R1, R0 ; R0 = 3*N/5
    LDR R2, MAddr ; R2 = &M (R2 points to M)
    STR R0, [R2] ; M = 3*N/5
    BX LR
    .endasmfunc
NAddr .field N,32
MAddr .field M,32
```



# Stack

PUSH {R0}

PUSH {R1}

PUSH {R2}

POP {R3}

POP {R4}

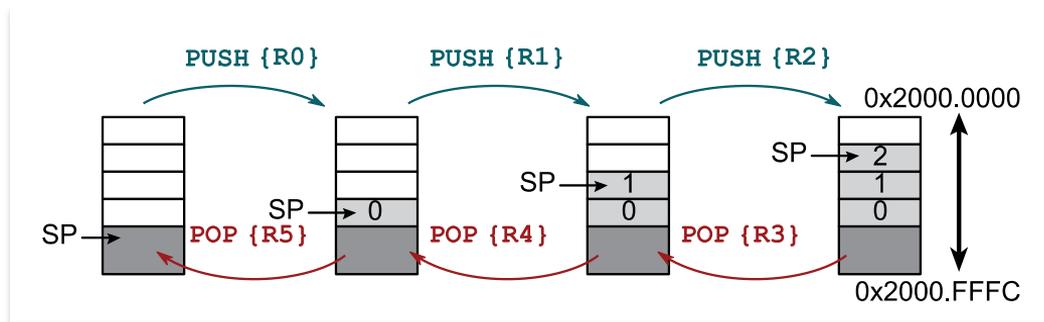
POP {R5}

## Push

1.  $SP=SP-4$
2. Store at SP

## POP

1. Read at SP
2.  $SP=SP+4$



## Usage

- Temporary storage
- Local variables

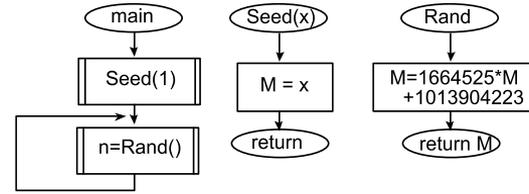


# Function calls

```

.data
.align 2
M .space 4
.text
.align 2
Seed: .asmfunc
LDR R1,MAddr ; R1=&M
STR R0,[R1] ; set M
BX LR
.endasmfunc
Rand: .asmfunc
LDR R2,MAddr ; R2=&M, address of M
LDR R0,[R2] ; R0=M, value of M
LDR R1,Slope
MUL R0,R0,R1 ; R0 = 1664525*M
LDR R1,Offst
ADD R0,R0,R1 ; 1664525*M+1013904223
STR R0,[R2] ; store M
LSR R0,#24 ; 0 to 255
BX LR
.endasmfunc
MAddr .field M,32
Slope .field 1664525,32
Offst .field 1013904223,32

```



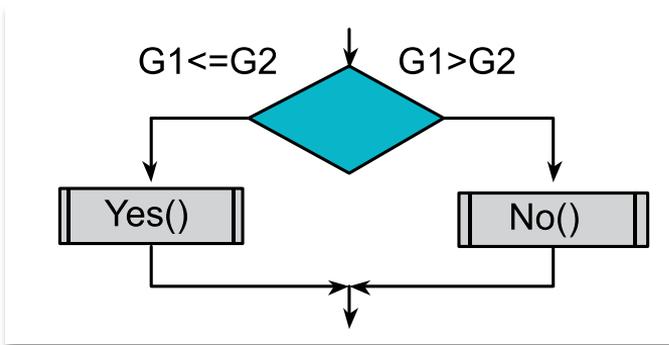
```

.data
.align 2
n .space 4
.text
.align 2
main: .asmfunc
MOV R0,#1
BL Seed
loop BL Rand
LDR R1,nAddr
STR R0,[R1]
B loop
.endasmfunc
nAddr .field n,32

```



# Conditionals



```

LDR R3,G2Addr ; R3=&G2, address of G2
LDR R2,[R3] ; R2=G2, value of G2
LDR R0,G1Addr ; R0=&G1, address of G1
LDR R1,[R0] ; R1=G1, value of G1
CMP R1,R2 ; compare G1 G2
BHI isNo
isYes BL Yes ; G1<=G2
      B done
isNo BL No
done

```

```

G1Addr .field G1,32
G2Addr .field G2,32

```

<b>Instruction</b>	<b>Branch if</b>
B target	; always
BEQ target	; equal (signed or unsigned)
BNE target	; not equal (signed or unsigned)
BLO target	; unsigned less than
BLS target	; unsigned less than or equal to
BHS target	; unsigned greater than or equal to
BHI target	; unsigned greater than
BLT target	; signed less than
BGE target	; signed greater than or equal to
BGT target	; signed greater than
BLE target	; signed less than or equal to

```

if (G2<=G1) {
    Yes ();
} else {
    No ();
}

```

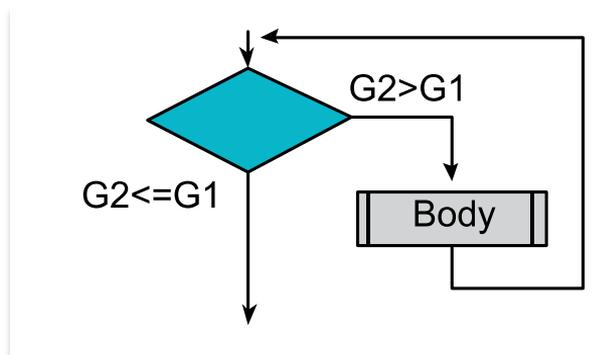
## Think of the three steps

- 1) bring first value into a register,
- 2) compare to second value,
- 3) conditional branch, bxx

(where xx is eq ne lo ls hi hs gt ge lt or le).  
The branch will occur if (first is xx second).



# While Loops



```
LDR R3,G2Addr ; R3=&G2, address of G2
LDR R2,[R3]   ; R2=G2, value of G2
LDR R0,G1Addr ; R0=&G1, address of G1
LDR R1,[R0]   ; R1=G1, value of G1
loop CMP R1,R2 ; compare G1 G2
      BLS done
      BL Body ; G1>G2
      B loop
done
```

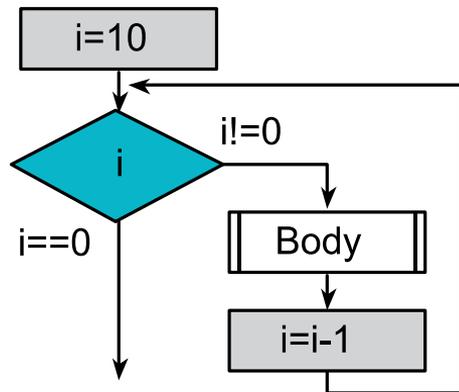
  

```
G1Addr .field G1,32 ;unsigned 32-bit number
G2Addr .field G2,32 ;unsigned 32-bit number
```

```
while (G2>G1) {
    Body();
}
```



# For Loops



```
for(i=10; i!=0; i--){  
    Body(); // 10 times  
}
```

```
loop    MOV    R4, #10  
        CMP    R4, #0  
        BEQ    done  
        BL    Body  
        SUB    R4, R4, #1  
        B     loop  
done
```

```
MOV    R4, #10  
loop   BL    Body  
        SUBS R4, R4, #1  
        BNE  loop
```

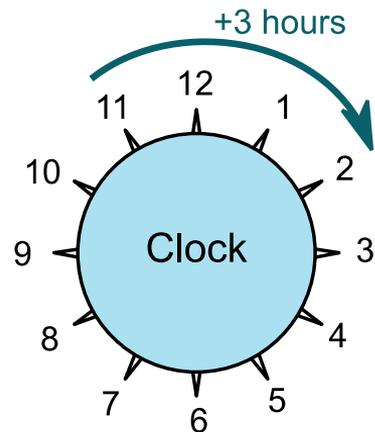


# ARM Cortex M Assembly Programming

## Summary

- Programming
  - Accessing memory
  - Logical and shift operations
  - Addition, subtraction, multiplication and divide
  - Stack
  - Functions, parameters
  - Conditionals
  - Loops

Register  
Immediate  
Indexed  
PC-relative



## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2018, Texas Instruments Incorporated