



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB3 ARMv7 Assembly Programming

2023 Fall
wangq9@mail.sustech.edu.cn



CONTENTS

1

Cortex-M3 introduction

2

CPUIlator system simulator

3

Basic assembly instructions

4

Practice



01

Cortex-M3 introduction



1. Cortex-M3 introduction

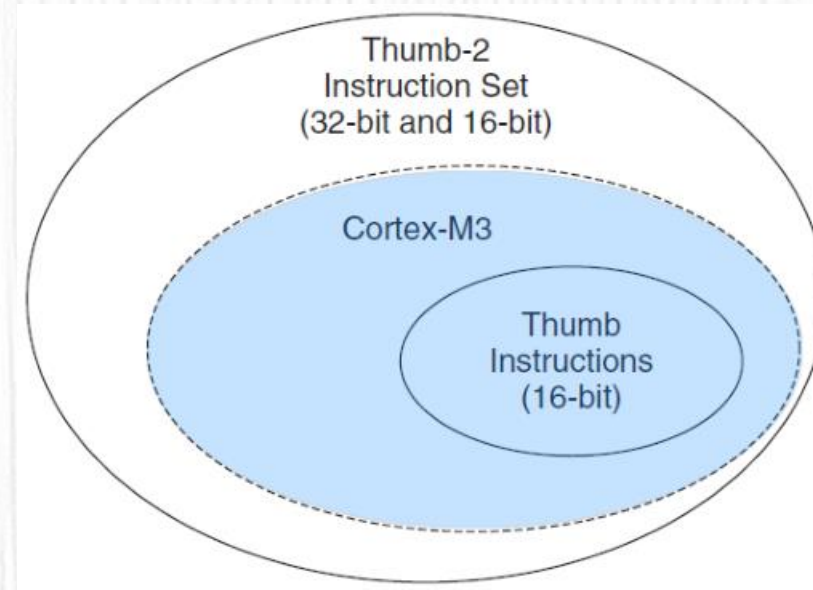
- **Instruction Set Development**
 - Historically (since ARM7TDMI), two different instruction sets are supported on the ARM processor: the ARM instructions that are 32-bit and Thumb instructions that are 16-bit.
 - During program execution, the processor can be dynamically switched between the ARM state or the Thumb state to use either one of the instruction sets.
 - The Thumb instruction set provides only a subset of the ARM instructions, but it can provide higher code density.
 - The Thumb instruction set is useful for products with tight memory requirements.

1. Cortex-M3 introduction

- The Thumb-2 Instruction Set Architecture (ISA)

- In 2003, ARM announced the Thumb-2 instruction set, which is a new superset of Thumb instructions that contains both 16-bit and 32-bit instructions.
- The Cortex-M3 supports only the Thumb-2 (and traditional Thumb) instruction set.
- With support for both 16-bit and 32-bit instructions in the Thumb-2 instructions set, there is no need to switch the processor between Thumb state (16-bit instructions) and ARM state (32-bit instructions).

The Relationship between Thumb-2 and Thumb Instruction Sets





1. Cortex-M3 introduction

- Registers (R0 to R15)

- General-Purpose Registers R0–R7 (low registers)
 - They can be accessed by all 16-bit Thumb and all 32-bit Thumb-2 instructions.
 - They are all 32-bit; the reset value is unpredictable.
- General-Purpose Registers R8–R12 (high registers)
 - They are accessible by all Thumb-2 but not by all 16-bit Thumb instructions.
 - These registers are all 32-bit; the reset value is unpredictable.
- R13: Stack Pointers
 - The Cortex-M3 contains two stack pointers, R13. They are banked so that only one is visible at a time.
- R14: The Link Register
 - When a subroutine is called, the return address is stored in the link register.
- R15: The Program Counter
 - The program counter is the current program address. This register can be written to control the program flow.

1. Cortex-M3 introduction

- Special Registers

- They have special functions and can be accessed only by special instructions.
- They cannot be used for normal data processing.
- The Cortex-M3 processor also has a number of special registers:
 - Program Status Registers (PSRs: APSR, IPSR, EPSR)
 - Interrupt Mask Registers (PRIMASK, FAULTMASK, BASEPRI)

Register	Function
xPSR	Provide ALU flags (Z, C), execution status, and current executing interrupt number
PRIMASK	Disable all interrupts except the nonmaskable interrupt (NMI) and HardFault
FAULTMASK	Disable all interrupts except the NMI
BASEPRI	Disable all interrupts of specific priority level or lower priority level
CONTROL	Define privileged status and stack pointer selection



02

CPUlator system simulator



2. CPULATOR system simulator

- CPULATOR is a simulator and debugger of a computer system (CPU, memory, and I/O devices) that runs inside a web browser.
- <https://cpulator.01xz.net/?sys=arm>
- Features
 - No download
 - No account sign up
 - Only Internet

- The simulator interface consists of the toolbar and a collection of movable panels.



- Step Into
F2

Step Over
Ctrl-F2

Step Out
Shift-F2

Continue
F3

Stop
F4


Restart
Ctrl-R

Reload
Ctrl-Shift-L


File ▾

Help ▾


Documentation


Documentation 


Feedback


Bugs, Feature requests 


Sample Programs

Infinite loop 

Audio output test 

Audio loopback test 

VGA test 

Video input test 

Version 811

Q Memory (Ctrl-M)

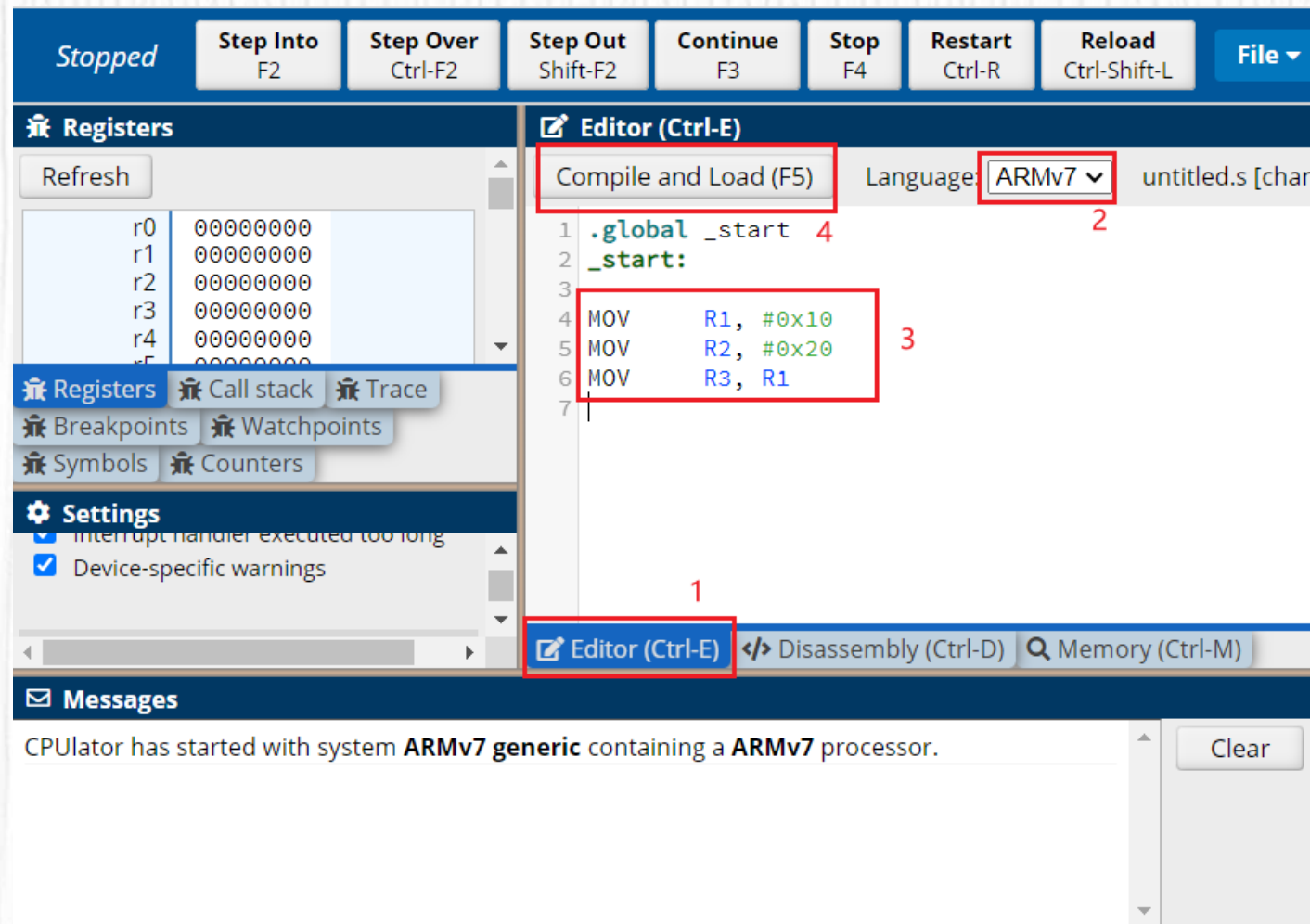
Go to address, label, or register:

Address	Memory contents and ASCII			
ffffffce0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffcf0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd00	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd10	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd20	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd30	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd40	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd50	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd60	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd70	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
ffffffd80	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa



2. CPULator system simulator

- Load assembly codes.



- Run the codes.

The screenshot shows the Visual Studio Code interface during a debugging session. The top toolbar indicates the program is 'Stopped'. The 'Step Into' button (F2) is highlighted with a red box. The 'Registers' panel on the left shows register r1 with value 00000010, also highlighted with a red box. The 'Disassembly' panel on the right shows assembly code, with the instruction 'mov r2, #32 ; 0x20' highlighted in yellow. The 'Settings' panel on the left shows 'Interrupt handler executed too long' and 'Device-specific warnings' checked.



03

Basic assembly instructions



3. Basic assembly instructions

- Bit Fields in Cortex-M3
Program Status Registers
 - N: Negative
 - Z: Zero
 - C: Carry / Borrow
 - V: Overflow

Registers		
Refresh		
r0	00000000	
r1	00000010	
r2	00000000	
r3	00000000	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000000	
cpsr	000001d3	NZCVI SVC
spsr	00000000	NZCVI ?

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q											
IPSR												Exception Number				
EPSR						ICI/IT	T				ICI/IT					

3. Basic assembly instructions

- Flag related instructions demo1

```
MOV    R6, #0x38
MOV    R7, #0x2F
ADDS   R6, R6,R7
```

Stopped

Step Into
F2

Step Over
Ctrl-F2

Step Out
Shift-F2

Continue
F3

Stop
F4

Restart
Ctrl-R

Reload
Ctrl-Shift-L

Registers

Refresh

r0	00000000
r1	00000010
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D)

Go to address, label, or register: 00000000

Address	Opcode	Disassembly
fffffffc	aaaaaaaa	bge
fffffffb	aaaaaaaa	bge
fffffffa	aaaaaaaa	bge
ffffff9f	aaaaaaaa	bge
ffffff9e	aaaaaaaa	bge
00000000	e3a06038	1 .global _start: 2 4 MOV _start: mov
00000004	e3a0702f	5 MOV R7, mov
00000008	e0966007	6 adds adds
0000000c	00000000	andeq andeq
00000010	aaaaaaaa	_end: bge
00000014	aaaaaaaa	bge
00000018	aaaaaaaa	bge

Stopped

Step Into
F2

Step Over
Ctrl-F2

Step Out
Shift-F2

Continue
F3

Stop
F4

Restart
Ctrl-R

Reload
Ctrl-Shift-L

Registers

Refresh

r0	00000000
r1	00000010
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000067
r7	0000002f
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000010
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D)

Go to address, label, or register: 00000010

Address	Opcode	Disassembly
fffffffc	aaaaaaaa	bge 0xfeaaaaac
00000000	e3a06038	1 .global _start 2 _start: 4 MOV R6, #0x38 _start: mov r6, #56 ; 0x38
00000004	e3a0702f	5 MOV R7, #0x2F mov r7, #47 ; 0x2f
00000008	e0966007	6 adds r6, r6, r7 adds
0000000c	00000000	andeq r0, r0, r0 andeq
00000010	aaaaaaaa	_end: bge 0xfeaaaaac0
00000014	aaaaaaaa	bge 0xfeaaaaac4
00000018	aaaaaaaa	bge 0xfeaaaaac8
0000001c	aaaaaaaa	bge 0xfeaaaaacc
00000020	aaaaaaaa	bge 0xfeaaaaad0

3. Basic assembly instructions

- Flag related instructions demo2

```
LDR    R0,=0x9C
LDR    R1,=0xFFFFFFFF64
ADDS   R0,R0,R1
```

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L

Registers Refresh

r0	00000000
r1	00000010
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000010

Address	Opcode
00000000	aaaaaaaa
00000004	aaaaaaaa
00000008	aaaaaaaa
0000000c	aaaaaaaa
00000010	e3a0009c
00000014	e3e0109b
00000018	e0900001
0000001c	00000000
00000020	00000000
00000024	00000000

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L

Execute until the current subroutine returns

Registers Refresh

r0	00000000
r1	ffffff64
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000010
cpsr	600001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000010

Address	Opcode	Disassembly
00000000	aaaaaaaa	bge 0xfeaaaa0
00000004	aaaaaaaa	bge 0xfeaaaa4
00000008	aaaaaaaa	bge 0xfeaaaa8
0000000c	aaaaaaaa	bge 0xfeaaaac
00000010	e3a0009c	1 LDR R0,=0x9C
00000014	e3e0109b	mov r0, #156
00000018	e0900001	2 LDR R1,=0xFFFFFFFF64
0000001c	00000000	mvn r1, #155
00000020	00000000	adds r0, r0, r1
00000024	00000000	andeq r0, r0, r0
00000010	aaaaaaaa	_end: bge 0xfeaaaac0
00000014	aaaaaaaa	bge 0xfeaaaac4
00000018	aaaaaaaa	bge 0xfeaaaac8
0000001c	aaaaaaaa	bge 0xfeaaaacc
00000020	aaaaaaaa	bge 0xfeaaaad0
00000024	aaaaaaaa	bge 0xfeaaaad4



3. Basic assembly instructions

- Flag related instructions demo2

LDR R0,=0x9C
LDR R1,=0xFFFFFFFF64
ADDS R0,R0,R1

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L

Registers Refresh

Register	Value
r0	00000000
r1	00000010
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000010

Address	Opcode	Disassembly
00000000	00000000	bge 0xfeaaaa0
00000004	00000000	bge 0xfeaaaa4
00000008	00000000	bge 0xfeaaaa8
0000000c	00000000	bge 0xfeaaaaac
00000010	00000000	1 LDR R0,=0x9C
00000014	00000000	mov r0, #156
00000018	00000000	2 LDR R1,=0xFFFFFFFF64
0000001c	00000000	mvn r1, #155
00000020	00000000	adds r0, r0, r1
00000024	00000000	andeq r0, r0, r0
00000028	00000000	_end:
0000002c	00000000	bge 0xfeaaaa0
00000030	00000000	bge 0xfeaaaa4
00000034	00000000	bge 0xfeaaaa8
00000038	00000000	bge 0xfeaaaaac
0000003c	00000000	bge 0xfeaaaa0
00000040	00000000	bge 0xfeaaaa4

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L

Execute until the current subroutine returns

Registers Refresh

Register	Value
r0	00000000
r1	ffffffff64
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000010
cpsr	600001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000010

Address	Opcode	Disassembly
00000000	00000000	bge 0xfeaaaa0
00000004	00000000	bge 0xfeaaaa4
00000008	00000000	bge 0xfeaaaa8
0000000c	00000000	bge 0xfeaaaaac
00000010	00000000	1 LDR R0,=0x9C
00000014	00000000	mov r0, #156
00000018	00000000	2 LDR R1,=0xFFFFFFFF64
0000001c	00000000	mvn r1, #155
00000020	00000000	adds r0, r0, r1
00000024	00000000	andeq r0, r0, r0
00000028	00000000	_end:
0000002c	00000000	bge 0xfeaaaa0
00000030	00000000	bge 0xfeaaaa4
00000034	00000000	bge 0xfeaaaa8
00000038	00000000	bge 0xfeaaaaac
0000003c	00000000	bge 0xfeaaaa0
00000040	00000000	bge 0xfeaaaa4

3. Basic assembly instructions

- Flag related instructions demo3

```
LDR    R0,=0xA5
LDR    R1,=0x23
SUBS   R0,R0,R1
```

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File

Registers Refresh

r0	00000000
r1	00000000
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000000

Address	Opcode
fffffffc	aaaaaaaa
00000000	e3a000a5
00000004	e3a01023
00000008	e0500001
0000000c	00000000
00000010	e5962004
00000014	e31204ff
00000018	0affffffc
0000001c	e31208ff
00000020	0affffffa
00000024	e5864008

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File

Registers Refresh

r0	00000082
r1	00000023
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000010
cpsr	200001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D) Go to address, label, or register: 00000000

Address	Opcode	Disassembly
fffffffc	aaaaaaaa	bge 0xfeaaaaac
1		.global _start
2		_start:
3		LDR R0,=0xA5
		_start:
00000000	e3a000a5	mov r0, #165 ; 0xa5
00000004	e3a01023	LDR R1,=0x23
00000008	e0500001	mov r1, #35 ; 0x23
0000000c	00000000	5 subs r0, r0, r1
		andeq r0, r0, r0
		_end:
00000010	e5962004	ldr r2, [r6, #4]
00000014	e31204ff	tst r2, #-16777216 ; 0
00000018	0affffffc	beq 0x10 (0x10: _end)
0000001c	e31208ff	tst r2, #16711680 ; 0
00000020	0affffffa	beq 0x10 (0x10: _end)
00000024	e5864008	str r4, [r6, #8]

3. Basic assembly instructions

- Flag related instructions demo4

LDR r3, =0xFFFF0000
LSLS r2, r3, #1

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File Help

Registers

Refresh

r0	00000000
r1	00000000
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D)

Go to address, label, or register: 00000000

Address

ffffffe8
ffffffec
fffffff0
fffffff4
fffffff8
fffffffc
00000000
00000004
00000008
0000000c

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File Help

Registers

Refresh

r0	00000000
r1	00000000
r2	fffe0000
r3	ffff0000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	0000000c
pc	00000008
cpsr	a00001d3 NZCVI SVC
spsr	a00001d3 NZCVI SVC

Disassembly (Ctrl-D)

Go to address, label, or register: 00000000

Address	Opcode	Disassembly
ffffffe8	aaaaaaaa	bge 0xfeaaaa98
ffffffec	aaaaaaaa	bge 0xfeaaaa9c
fffffff0	aaaaaaaa	bge 0xfeaaaaa0
fffffff4	aaaaaaaa	bge 0xfeaaaaa4
fffffff8	aaaaaaaa	bge 0xfeaaaaa8
fffffffc	aaaaaaaa	bge 0xfeaaaaac
1 .global _start		
2 _start:		
4 LDR r3, =0xFFFF0000		
_start:		
00000000	e51f3000	ldr r3, [pc, #-0] ; 0x8
00000004	e1b02083	lsls r2, r3, #1
4 LDR r3, =0xFFFF0000		
00000008	ffff0000	svc #16711680 ; 0xff0000
0000000c	00000000	andeq r0, r0, r0
_end:		



3. Basic assembly instructions

- Flag related instructions demo5

LDR r2 ,= 0x00000001

LSRS r3, r2, #1

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File

Registers Disassembly (Ctrl-D)

Refresh

r0	00000000
r1	00000000
r2	00000000
r3	00001234
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	0000000c
pc	00000000
cpsr	000001d3 NZCVI SVC
spsr	600001d3 NZCVI SVC

Go to address, label, or register:

Address	Opcode
ffffffe8	aaaaaaaa
ffffffec	aaaaaaaa
fffffff0	aaaaaaaa
fffffff4	aaaaaaaa
fffffff8	aaaaaaaa
fffffffc	aaaaaaaa
00000000	e3a02001
00000004	e1b030a2
00000008	fffff000
0000000c	00000000
00000010	aaaaaaaa

Stopped Step Into F2 Step Over Ctrl-F2 Step Out Shift-F2 Continue F3 Stop F4 Restart Ctrl-R Reload Ctrl-Shift-L File

Registers Disassembly (Ctrl-D)

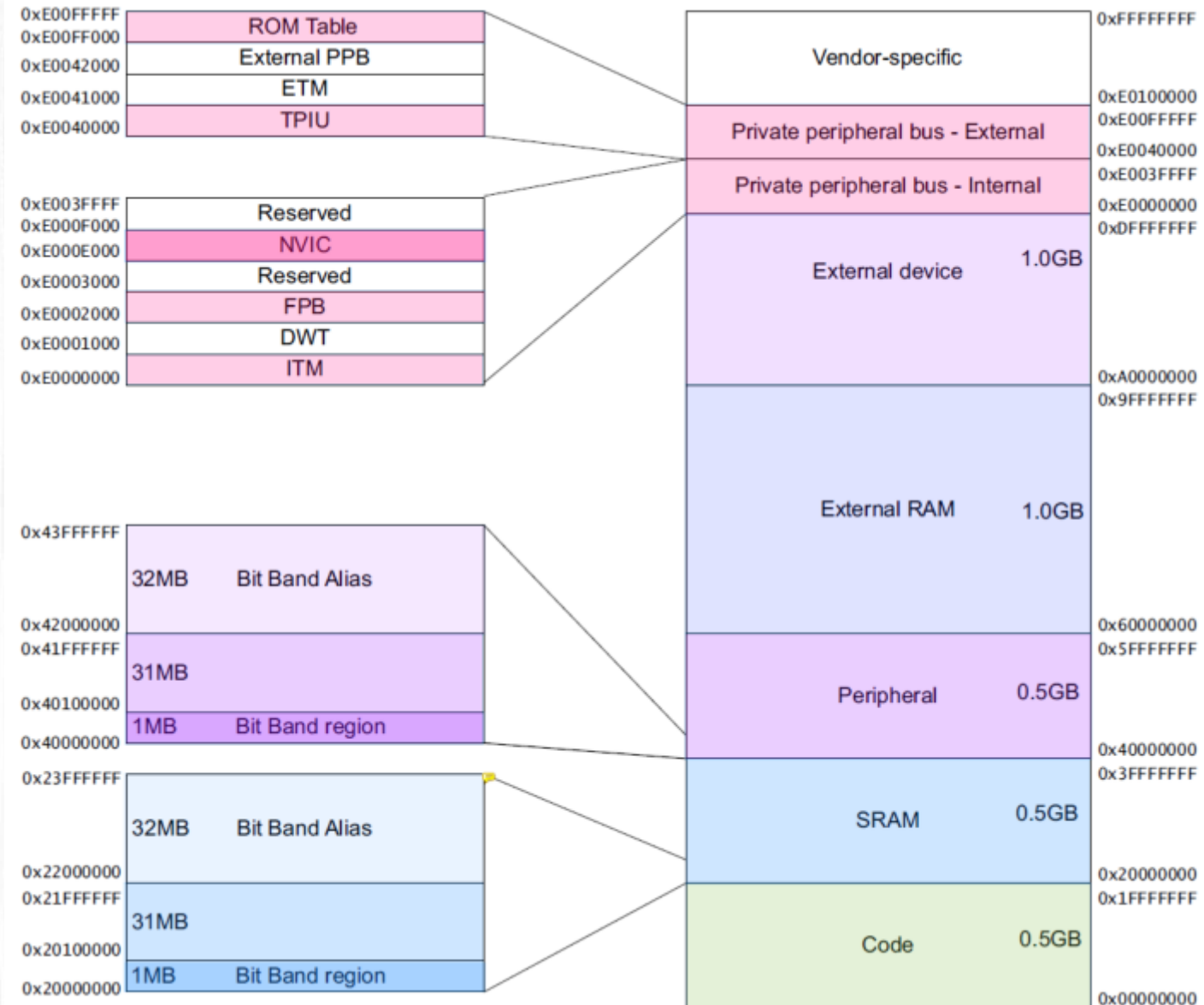
Refresh

Go to address, label, or register: 00000000

Address	Opcode	Disassembly
ffffffe8	aaaaaaaa	bge 0xfeaaaa98
ffffffec	aaaaaaaa	bge 0xfeaaaa9c
fffffff0	aaaaaaaa	bge 0xfeaaaaa0
fffffff4	aaaaaaaa	bge 0xfeaaaaa4
fffffff8	aaaaaaaa	bge 0xfeaaaaa8
fffffffc	aaaaaaaa	bge 0xfeaaaaac
1 .global _start		
2 _start:		
4 LDR r2 ,= 0x00000001		
_start:		
00000000	e3a02001	mov r2, #1 ; 0x1
00000004	e1b030a2	lsrs r3, r2, #1
_end:		
00000008	fffff000	svc #16711680 ;
0000000c	00000000	andeq r0, r0, r0
00000010	aaaaaaaa	bge 0xfeaaaaac0

3. Basic assembly instructions

- The Cortex-M3 processor has a total of 4 GB of address space.
- Program code can be located in the Code region, the SRAM region, or the External RAM region.
- Another 0.5 GB block of address range is allocated to on-chip peripherals.
- Two slots of 1 GB memory space are allocated for external RAM and external devices.
- The last 0.5 GB of memory is for the system-level components, internal peripheral buses, external peripheral bus, and vendor-specific system peripherals.





3. Basic assembly instructions

- Common memory access instructions

Example	Description
LDRB Rd, [Rn, #offset]	Read byte from memory location Rn+offset
LDRH Rd, [Rn, #offset]	Read half-word from memory location Rn+offset
LDR Rd, [Rn, #offset]	Read word from memory location Rn+offset
LDRD Rd1, Rd2, [Rn, #offset]	Read double word (64-bits) from memory location Rn+offset, set Rd1 as lower 32-bits and Rd2 as higher 32-bits
STRB Rd, [Rn, #offset]	Store byte to memory location Rn+offset
STRH Rd, [Rn, #offset]	Store half-word to memory location Rn+offset
STR Rd, [Rn, #offset]	Store word to memory location Rn+offset
LDRD Rd1, Rd2, [Rn, #offset]	Store double word to memory location Rn+offset



3. Basic assembly instructions

- Memory access instruction demo

```
.global _start
_start:
    LDR R0,=list
    LDR R1,[R0]
    LDR R2,[R0, #4]
    ADD R2, R1, R2

.data
list:
    .byte 0x11, 0x22, 0x33, 0x44
    .word 0x11223344, 0x55667788
```

Read this piece of codes, tell the result of R2, describe how to load data from memory?

Q Memory (Ctrl-M)				
Go to address, label, or register: <input type="text"/>				
Address	Memory contents and ASCII			
00000000	e59f0008	e5901000	e5902004	e0812002
00000010	00000018	00000000	44332211	11223344
00000020	55667788	00000000	00000000	00000000
00000030	00000040	00000000	00000000	00000000

Editor (Ctrl-E) Disassembly (Ctrl-D) Q Memory (Ctrl-M)

- After loading, we can observe data in memory window.
- Q1: What is the starting address of list?
 - Q2: Does byte 0x11, 0x22, 0x33, 0x44 and word 0x11223344 be stored the same in memory?
 - Q3: Does ARMv7 adopt large endianness or small endianness storage mode?

3. Basic assembly instructions

- Common conditional branch instructions

Instruction	Function
B	Branch
BL	Branch and link
BEQ	Branch if two values are equal
BNE	Branch if two values are not equal
BCC	Branch if Carry = 0
BCS	Branch if Carry = 1
TBB	Table branch byte; forward branch using a table of single byte offset
TBH	Table branch half word; forward branch using a table of half word offset



3. Basic assembly instructions

- Conditional branch instruction demo

```
.global _start
_start:
    MOV R1, #0xa
    MOV R2, #0xb
    CMP R1, #0x1
    BNE else    //go to else if _?_
then:
    MOV R2, #3    //R2 = 3
    B    endif    //go to endif
else:
    MOV R2, #4    //b = 4
endif:
    LDR R1, =dst
    STR R2, [R1]
dst:
    .asciz "00000000"
```

Read this piece of codes, tell the results of following questions.

- Q1: What condition(s) should be filled in blank `_?_` according to the codes?
- Q2: What are the address of labels then, else, endif and dst individually?
- Q3: Which address will be accessed after BNE instruction?
- Q4: What value will be stored in memory?
- Q5: If we want to store another value into memory, how should we change the codes?



3. Basic assembly instructions

- Conditional branch instruction demo -- loop

```
.global _start
_start:
    MOV r0, #0 // i
    MOV r1, #0 // sum

loop:
    CMP r0, #10 // check whether i < 10
    BGE endloop // skip if _?_
    ADD r1, r1, r0 // sum += i
    ADD r0, r0, #1 // i++
    B loop
endloop:
```

Read this piece of codes, tell the results of following questions.

- Q1: What condition(s) should be filled in blank `_?_` according to the codes?
- Q2: How many times did the loop code block execute?
- Q3: What's the function of the codes?
- Q4: What are the final values of R0, R1?



04

Practice

4. Practice

- 1. Write your own codes and test in emulator, which sets C=1, and V=1;
- 2. Read the codes on right
 - Answer the questions 1~4
 - What's the usage of the code?
 - **NOTES:** this piece of codes **can not** run in CPUlator.

```
1  RCC_APB2ENR EQU 0x40021018 ; set RCC_APB2ENR as constant 0x40021018
2  GPIOA_CRL EQU 0x40010800
3  GPIOA_CRH EQU 0x40010804
4  GPIOA_IDR EQU 0x40010808
5  GPIOA_ODR EQU 0x4001080C
6  EXPORT __main
7  AREA MAIN, CODE, READONLY
8  __main
9      LDR R1,=RCC_APB2ENR
10     LDR R0,[R1]
11     ORR R0,R0,#0xFC
12     STR R0,[R1] ;__1. Purpose of the previous lines?__
13     LDR R1,=GPIOA_CRH
14     LDR R0,=0x44444443
15     STR R0,[R1] ;__2. Purpose of the previous lines?__
16     LOOP:
17     LDR R1,=GPIOA_ODR
18     LDR R0,[R1]
19     EOR R0,R0, 0x00000100
20     STR R0,[R1] ;__3. Purpose of the previous lines?__
21     BL delay ;jump to line 24 to execute function delay
22     B LOOP
23 delay:
24     MOV R4, #0
25     MVN R5, #1 ; __4. value of R5 = ?__
26 d_loop:
27     ADD R4, R4, #1
28     CMP R5, R4
29     BGE d_loop ; if >=, jump to d_loop
30     BX LR ; jump to line 22
31 END
```