UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Junior Sophister BAI **Trinity Term 2014**

Microprocessor Systems I

Thursday, 15th May 2014 **Luce Upper** **14:00 - 16:00**

Prof. John Waldron

Instructions to Candidates

Question 1 is worth 50 marks. Each part of Question 2 is worth 5 marks, your best ten answers are counted. Answer both questions. Please detach the last page of the exam booklet and mark your answers on this and include with your answer book.

To be accompanied by an ARM Instruction Set and Addressing Mode Summary booklet.

Permitted Materials
Non-programmable calculators are permitted for this examination.

Section A

In this section marks are awarded for neatness, organisation, spelling, ability to communicate technical information and results, as well as assembly programming syntax, commenting and skill.

Design and write an ARM Assembly Language program that will convert an ASCII string representation of a decimal number stored in memory into a 32 bit 2s complement binary version, which you will also store in memory. Negative numbers will be indicated by a - at the start of the number. Your solution should work with both + and - at the start of the string, so for example -34 would be stored as 0xFFFFFFDE.

1. (a) Describe what you are attempting to do in English. (10 marks)

   (b) Outline your algorithm using diagrams and pseudo code as appropriate. (15 marks)

   (c) Write down the actual ARM assembly code you would use, including comments. (15 marks)

   (d) Explain the test cases you would use, why you would chose them and the results expected. (10 marks)

Section B

Question 2.1

```
;;
;; After execution of the following instructions
;; what value will be in register r11?
;;
        MOV   r0, #0x6
        MOV   r1, #0x6
        ADD   r11, r1, r0
```

(A) 0x0000000C (B) 0x00000001 (C) 0x00000010
(D) 0x00000013 (E) 0x0000000D (F) OTHER (5 marks)

Question 2.2

```
;;
;; After execution of the following instructions
;; what value will be in the condition code flags?
;;
        MOV   r0, #0xC0000000
        MOV   r1, #0xF0000000
        SUBS  r3, r1, r0
```

(A) 0x5 (B) 0x3 (C) 0x1
(D) 0x7 (E) 0x2 (F) OTHER (5 marks)

Question 2.3

```
;;
;; After execution of the following instructions
;; what value will be in the condition code flags?
;;
        MOV   r0, #0xFFFFFFFC
        MOV   r1, #0x20
        SUBS  r2, r1, r0
```

(A) 0x5 (B) 0x7 (C) 0x6
(D) 0x0 (E) 0xA (F) OTHER (5 marks)

Question 2.4

```
;;
;; After execution of the following instructions
;; what value will be in register r4?
;;
        LDR   r0, =0x7937
        LDR   r1, =0xA89E
        EOR   r4, r1, r0, ROR #6
```

(A) 0x00000005 (B) 0x00000004 (C) 0x3120BE1A
(D) 0xDC00A97A (E) 0x78998895 (F) OTHER (5 marks)

```
Question 2.5
;;
;; After execution of the following instructions
;; what value will be in register r1?
;;
        LDR    r0, =0xEB
        LDR    r1, =0x7D
        CMP    r0, r1
        BGE    a_label
        SUBS   r1, r1, #0xA7
a_label
        BLO    end
        SUBS   r1, r1, #0xA
end
```

(A) 0x000000AE (B) 0x00000889 (C) 0x00000073
(D) 0x0000009F (E) 0x00000001 (F) OTHER (5 marks)

```
Question 2.6
;;
;; After execution of the following instructions
;; what value will be in register r3?
;;
        LDR    r0, =eoa
        LDR    r1, =arr
        MOV    r3, #0
loop    LDRB   r2, [r1], #1
        ADD    r3, r2, r3
        CMP    r1, r0
        BNE    loop

arr     DCB    0x3B, 0x28, 0x55
eoa
```

(A) 0x000000B8 (B) 0x00007F38 (C) 0x00002B20
(D) 0x0000011D (E) 0x00000001 (F) OTHER (5 marks)

```
Question 2.7
;;
;; After execution of the following instructions
;; what value will be in register r1?
;;
        LDR     r0, =testcase
        MOV     r1, #0
loop    LDRB    r2, [r0]
        CMP     r2, #'Z'
        BLO     skip
        ADD     r1, r1, #1
skip    ADD     r0, #1
        CMP     r2, #0
        BNE     loop

testcase
        DCB     "uhalya16oG",0
```

(A) 0x00000008 (B) 0x00000002 (C) 0x00000005
(D) 0x00000007 (E) 0x00000003 (F) OTHER (5 marks)

```
Question 2.8
;;
;; After execution of the following instructions
;; what value will be in register r1?
;;
        LDR    r0, =nums
        MOV    r1, #0
        LDRB   r2, [r0], #2
        ADD    r1, r2
        LDRB   r2, [r0], #2
        ADD    r1, r2
        LDRB   r2, [r0, #2]
        ADD    r1, r2
        LDRB   r2, [r0, #-1]!
        ADD    r1, r2
        LDRB   r2, [r0, #-2]!
        ADD    r1, r2

nums    DCB    0x7, 0x5, 0xE, 0x5
        DCB    0x1, 0xD, 0x5, 0xB
```

(A) 0x0000001A (B) 0x00000024 (C) 0x00000001
(D) 0x000000B4 (E) 0x00000038 (F) OTHER (5 marks)

Question 2.9

```
;;
;; After execution of the following instructions
;; what value will be in register r4?
;;
        MOV     r1, #0
        LDR     r0, =str
        LDR     r3, =0xA0000000
loop    LDRB    r2, [r0], #1
        STRB    r2, [r3], #1
        ADD     r1, r1, #1
        CMP     r2, #0
        BNE     loop
        LDR     r3, =0xA0000000
        LDRB    r4, [r3, #3]

str     DCB     "R447Ghz",0
```

(A) 0x0000005D (B) 0x00000005 (C) 0x0000000B
(D) 0x00000339 (E) 0x00000037 (F) OTHER (5 marks)

Question 2.10

```
;;
;; After execution of the following instructions
;; what value will be in register r0?
;;
        LDR     r12, =0xA4000000
        LDR     r0, =0x34
        LDR     r1, =0x44
        LDR     r2, =0x3E
        LDR     r5, =0x9
        STMDB   r12!, {r0-r2, r5}
        LDR     r0, [r12, #8]
        SUB     r0, #0x2F
```

(A) 0x0000000C (B) 0x0000000B (C) 0x0000000A
(D) 0x0000000F (E) 0x00000016 (F) OTHER (5 marks)

Question 2.11

```
;;
;; After execution of the following instructions
;; what value will be in register r0?
;;
        MOV     r1, #'8'
        BL      vp
        MOV     r1, #'M'
        BL      vp
stop    B       stop
vp      MOV     r0, #0
        CMP     r1, #'a'
        BCC     yes
        BX      lr
yes     MOV     r0, #1
        BX      lr
```

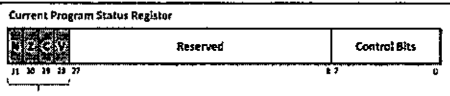
(A) 0x0000000A (B) 0x00000001 (C) 0x00000006
(D) 0x00000009 (E) 0x00000004 (F) OTHER (5 marks)

Question 2.12

```
;;
;; After execution of the following instructions
;; what value will be in register r5?
;;
        LDR     sp, =0xA4000000
        LDR     r0, =0xC2
        LDR     r1, =0x37
        LDR     r2, =0xCD
        STR     r0, [sp, #-4]!
        STR     r1, [sp, #-4]!
        STR     r2, [sp, #-4]!
        BL      vp
        ADD     sp, #12
        SUB     r5, r5, r0
stop    B       stop
vp      STMFD    sp!,{r0-r4,lr}
        LDR      r0, [sp, #8+24]
        LDR      r3, [sp, #4+24]
        LDR      r1, [sp, #0+24]
        ADD      r0, r1, r3
        ADD      r5, r1, r3
        LDMFD    sp!,{r0-r4,pc}
```

(A) 0x00000042 (B) 0x0000039C (C) 0x00000032
(D) 0x00000070 (E) 0x00000001 (F) OTHER (5 marks)

Condition Code Flags



ASCII Table

	0	1	2	3	4	5	6	7
0	NUL	DLE	SPACE	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Conditional Branch Instructions

Branch Instruction	Condition Code Flag Evaluation	Description
B (or BAL)	don't care	unconditional (branch always)
BEQ	Z	equal
BNE	$\bar{Z}$	not equal
BCS / BHS	C	unsigned ≥
BCC / BLO	$\bar{C}$	unsigned <
BMI	N	negative
BPL	$\bar{N}$	positive or zero
BVS	V	overflow
BVC	$\bar{V}$	no overflow
BHI	$C\bar{Z}$	unsigned >
BLS	$\bar{C} + Z$	unsigned ≤
BGE	$NV + \bar{N}\bar{V}$	signed ≥
BLT	$N\bar{V} + \bar{N}V$	signed <
BGT	$\bar{Z}(NV + \bar{N}\bar{V})$	signed >
BLE	$Z + N\bar{V} + \bar{N}V$	signed ≤

Summary of LDR/STR Addressing Modes

Addressing mode	Syntax	W, B	H, SH, SB	Operation
Immediate Offset	`[<Rn>, #+/-<offset>]`	✓	✓	address ← Rn +/- offset
Register Offset	`[<Rn>, +/-<Rm>]`	✓	✓	address ← Rn +/- Rm
Scaled Register Offset	`[<Rn, +/-<Rm>, <shift> #<count>]`	✓		address ← Rn +/- (Rm <shift> <count>)

Immediate Pre-Indexed	`[<Rn>, #+/-<offset>]!`	✓	✓	Rn ← Rn +/- offset
				address ← Rn

Register Pre-Indexed	`[<Rn>, +/-<Rm>]!`	✓	✓	Rn ← Rn +/- Rm
				address ← Rn

Scaled Register Pre-Indexed	`[<Rn, +/-<Rm>, <shift> #<count>]!`	✓		Rn ← Rn +/- (Rm <shift> <count>)
				address ← Rn

Immediate Post-Indexed	`[<Rn>], #+/-<offset>`	✓	✓	address ← Rn
				Rn ← Rn +/- offset

Register Post-Indexed	`[<Rn>], +/-<Rm>`	✓	✓	address ← Rn
				Rn ← Rn +/- Rm

Scaled Register Post-Indexed	`[<Rn], +/-<Rm>, <shift> #<count>`	✓		address ← Rn
				Rn ← Rn +/- (Rm <shift> <count>)

Exam Number	Seat Number		
Question 2.1	Question 2.2	Question 2.3	Question 2.4
Question 2.5	Question 2.6	Question 2.7	Question 2.8
Question 2.9	Question 2.10	Question 2.11	Question 2.12