National Sun Yat-Sen University ASSEMBLY LANGUAGE AND MICROCOMPUTER Final Exam 9:20-11:30AM

Name:

Note: Although there are more than 100 points for this exam, the maximum score you can get is 100 points.

- 1. Refer to the following 3-stage (fetch, decode, execute) ARM7 pipeline data path. (10 pts)
 - (a) Find out the number of cycles it will takes to run the ARM instruction <u>STR r0, [r1], #4</u> at the execution stages. <u>(4 pts)</u>
 - (b) Show the datapath activity at each cycle. (6 pts)
- 2. Show how the variable *data* is organized in the little-endian memory without packing. (10 pts)

struct T1 {float a[2], short b};
struct T2 { T1 c , char d2[5] } data;



- 3. Answer the following short questions: (9 pts)
 - (a) Explain what "callee-saved" register variables mean. (3 pts)
 - (b) Explain what the "**stack**" is and list three different usages of stack to support the high-level programming. <u>(6 pts)</u>
- 4. For the following simple assembly code: (15 pts)
 - (a) Explain the function of this code. (5 pts)
 - (b) Describe the main drawback of this realization. Rewrite and improve this code to solve the drawback. (*10 pts*)

	BL	TAB
TAB	CMP	r0, #0
	BEQ	SUB0
	CMP	r0, #1
	BEQ	SUB1
	•••••	•••••
	CMP	r0, #9
	BEQ	SUB9

- 5. For the 32-bit ARM instruction set: (20 pts)
 - (a) Write a C subroutine that matches the code shown in the right. (8 ts)
 - (b) Write an ARM program to implement a positive integer division subroutine *div(int A, int D, int* R)* which returns the quotient of *A* divided by *D*. The remainder of this division has to be written to the memory location pointed by *R*. You should try to use as fewer instructions as possible. (12 ts)

SubRt		
	MOV	r2, r0;
	MOV	r0, #0;
	MOV	r1, #0x10;
LOOP	LDR	r3, [r2], #4
	SUBS	r1, r1, #1
	ADD	r0, r3, r0
	BNE	LOOP
	BX	r14

- 6. The instruction coding of Thumb data processing instructions is shown in the following figure.
- (a) Check if the following Thumb instruction syntax is correct. If not, you should also explain why.
 - (10 pts)
 - (1) ADD r3, r8, r2
 - (2) SUBNE r2, #7
 - (3) CMP r4, r9
 - (4) POP r3, lr
 - (5) BLGT Subroutine (* Subroutine represents a symbol/label in the program)

(b) Write the equivalent 32-bit ARM instruction for the following Thumb instruction: (9 pts)

- (1) PUSH r4, r0, lr
- (2) SUB r3, #62
- (3) LSR r2, r3

15		10	9	8		6	5		3	2	0	_
0001	10		А	F	٦m		F	Rn		R	d	
		4.0				•	_		_	_	-	
15		10	9	8		6	5		3	2	0	٦.
0001	А	#imm3			Rn			Rd				
15 13	12 11	10		8	7						0	
001	Ор	Rd	/Rn	າ #imm8								
<u>15 13</u>	12 11	10				6	5		3	2	0	
000	Ор		#sh				Rn Rd					
												1000000
15		10	9			6	5		3	2	0	
15 0 1 0 0	0 0	10	9	0	p	6	5 Rm	/Rs	3	2 Rd/	0 Rn]
15 0 1 0 0 15	00	<u>10</u> 10	9	0	р 7	6	5 Rm 5	/Rs	3	2 Rd/ 2	0 Rn 0	
15 0 1 0 0 15 0 1 0 0	00	<u>10</u> 10	9 9 0	0 8 0	p 7 D	6 6 M	5 Rm 5 F	/Rs Rm	3	2 Rd/ 2 Rd/	0 Rn 0 Rn	
15 0 1 0 0 15 0 1 0 0 15	00	10 10 10	9 9 O I	0 8 0 8	p 7 D 7	6 6 M	5 Rm 5 F	/Rs Rm	3	2 Rd/ 2 Rd/	0 Rn 0 Rn 0	
15 0 1 0 0 15 0 1 0 0 15 1 0 1 0	0 0 0 1 12 11 R	10 10 10	9 9 Ol	0 8 0 8	р 7 D 7	6 6 M	5 Rm 5 F	/Rs Rm	3 3 18	2 Rd/ 2 Rd/	0 Rn 0 Rn 0	
15 0 1 0 0 15 15 1 0 1 0 0 15 15	0 0 0 1 12 11 R	10 10	9 9 Of Rd	0 8 0 8 8	p 7 D 7 7	6 6 M	5 Rm 5 F	/Rs Rm	3 3 18	2 Rd/ 2 Rd/	0 Rn 0 Rn 0	

- (1) ADD | SUB Rd, Rn, Rm
- (2) ADD|SUB Rd,Rn,#imm3
- (3) <Op> R d/Rn ,#imm8
- (4) LSL|LSR|ASR Rd,Rn,#shift
- (5) <Op> Rd/Rn,Rm/Rs
- (6) ADD | CMP | MOV Rd/Rn, Rm
- (7) ADD Rd, SP | PC, #imm8
- (8) ADD | SUB SP, SP, #imm7

- 7. Find out the 32-bit instruction coding for the following ARM instructions based on the given coding information. (12 pts)
 - (a) STRB r7, [r1, r0, LSL #2]
 - (b) LDREQ r4, [r2, #-8]!
 - (c) LDR r1, [r9], #4.

			Coding	table of Al	RM condit	tion co	odes			
0000	EQ	0100	MI	1000	HI	110	O GT			
0001	NE	0101	PL	1001	LS	110	1 LE			
0010	CS/HS	0110	VS	1010	GE	1110	O AL			
0011	CC/LO	0111	VC	1011		111	1 NV			
Coding table of Shift Operation										
00	LSL	01	LSR	10	ASR	11	ROR			
00 The load an u or w A pre addre comp load then, reque the b comp load then, reque the b comp load then, reque to the assen i i i i i i i i i i i i i i i i i i i	LSL 31 cond address is used to (L=1) or store (L= insigned byte (B==) or (B=0) -indexed(P=1) assing mode uses is buted address for to or store operation, when write-back asser egister to the buted value e 7(b) show e original C ably code by finclude <stdi nt func1 (int nt func1 (int nt main() int a, printf return nt func1 (int int c[b = c[</stdi 	01 28 27 26 25 242 0 1 # P u 1 25 0 - 25 0 - 25 0 - 25 0 - 25 1 - 25 25 25 25 25 25 25 25 25 25	LSR 3 22 21 20 19 J B W L F immedi al assembl wn in Fig ten space 20]; 5); s %d %d\n",	ate shift length shift type offset register y code corri g. 7(a). Corri regions. <u>(15</u>	ASR ASR 2 11 ase register oad/store write-back (auta insigned byte// information of the second pre-/post-index 11 12-bi 11 7 #shift DRLS pc, responding mplete the 5 pts)	t immed fr 11 offset ion regis vord t immed fr 1, r(fur sa t fur sa sd	ROR 0 0 ter Adding (U subtractir unsigned or registe 0 iate 0 adding (U subtractir unsigned or registe 0 adding (U subtractir unsigned or registe 0 adding (U subtractir 0 iate 0 cr 0 cr 0 xt 0x000000000 0x0000000010 0x000000012 0x00000012 0x00000020 0x00000021 0x00000022 0x00000032 0x00000032 0x00000032 0x00000032 0x00000044 0x00000050 0x00000051 0x00000052 0x00000	as starting = 1) or g (U=0) ar immediate r offset	.35 r13, r13, #0x28 r0, r1, r0 	
	return	(b):					0x00000064:	DCD	544417056	
}	iciuili	(5),					0x0000006c:	DCD	2660	