NAME: _____

SANTA CLARA UNIVERSITY
Department of Computer Engineering

COEN 020 Midterm Exam Spring 2017

(Closed book & notes; No electronic devices)

Time Allowed: 1 hour

SCU's Academic Integrity Pledge

"I am committed to being a person of integrity. I pledge, as a member of the Santa Clara University community, to abide by and uphold the standards of academic integrity contained in the Student Conduct Code."

Signature:_____

Your Points:	
Max Points:	116
Your Score:	%

Points this page: _____

1. [4 pts] What is the maximum value that can be represented as an N-bit unsigned number?

   a. 2^N b. 2^{N-1} **c. 2^N-1** d. $2^{N-1}-1$

2. [4 pts] What is the most negative value that can be represented as an N-bit 2's complement number?

   a. -2^N **b. -2^{N-1}** c. $-(2^N-1)$ d. $-(2^{N-1}-1)$

3. [5 pts] Convert the unsigned decimal value 33_{10} to radix 7.

   33 ÷ 7 → Q=4, R=5 → 5.

   4 ÷ 7 → Q=0, R=4 → **45.**

4. [5 pts] Convert the unsigned radix 3 value 212.2_3 to base 9.

   212.2_3 → 2 12 . 2 → 02 12 . 20 → **25.6_9**

5. [5 pts] Convert signed 2's complement value 1000.0110_2 to decimal.

   $1000.0110_2 = -8 + .25 + .125 =$ **-7.625_{10}**

6. [5 pts] Convert -20_{10} to an 8-bit 2's complement representation.

   $+20 = 16 + 4 = 00010100_2$ → -20 = **11101100_2**

Points this page: _____

7. [5 pts ea] Look at the C code on the left, then circle the assembly language version that is correct.

int32_t a, b ;
 	
a = b + 5 ;
	LDR R0,a
	LDR R1,b
	ADD R0,R1,5
		LDR R0,b+5
		STR R0,a
			LDR R0,b
			ADD R2,R0,5
			STR R2,a

int32_t f1(int32_t s32)
{
int32_t f2(int32_t) ;
 	
return s32 + f2(s32) ;
}
	f1: LDR R0,s32
	BL f2
	ADD R0,R0,s32
	BX LR
		f1: PUSH {R4,LR}
		MOV R4,R0
		BL f2
		ADD R0,R0,R4
		POP {R4,PC}
			f1: PUSH {LR}
			MOV R1,R0
			BL f2
			ADD R0,R0,R1
			POP {LR}
			BX LR

void f3(int16_t a16[ ])
{
a16[5] = 0 ;
}
	f3: LDR R1,=0
	ADR R0,a16
	STRH R1,[R0,10]
	BX LR
		f3: LDR R1,=0
		STRH R1,a16+10
		BX LR
			f3: LDR R1,=0
			STRH R1,[R0,10]
			BX LR

void f4(int64_t *p64)
{
*p64 = -5 ;
}
	f4: LDRD R1,R2,=-5
	STRD R1,R2,[R0]
	BX LR
		f4: LDR R1,=-5
		LDR R2,=-1
		STRD R1,R2,[R0]
		BX LR
			f4: LDR R1,=-5
			LDR R2,=0
			STRD R1,R2,[R0]
			BX LR

uint8_t f5(uint8_t u8)
{
if (u8 != 0) --u8 ;
return u8 ;
}
	f5: CMP R0,0
	BNE L1
	L1: SUB R0,R0,1
	BX LR
		f5: CMP R0,0
		BNE L1
		SUB R0,R0,1
		L1: BX LR
			f5: CMP R0,0
			BEQ L1
			SUB R0,R0,1
			L1: BX LR

uint8_t f6(uint8_t u8)
{
if (u8 != 0) --u8 ;
else u8 = 100 ;
return u8 ;
}
	f6: CMP R0,0
	ITE NE
	SUB R0,R0,1
	LDR R0,=100
	BX LR
		f6: CMP R0,0
		IT NE
		SUBNE R0,R0,1
		LDREQ R0,=100
		BX LR
			f6: CMP R0,0
			ITE NE
			SUBNE R0,R0,1
			LDREQ R0,=100
			BX LR

Points this page: _____

8. [5 pts ea] Look at the C code on the left, then circle the assembly language version that is correct.

int64_t f7(int64_t a, int32_t b)
{
return a - b ;
}
	f7: SUB R0,R0,R2
	BX LR
		f7: SUBS R0,R0,R2
		SBC R1,R1,0
		BX LR
			**f7: ASR R3,R2,31
			SUBS R0,R0,R2
			SBC R1,R1,R3
			BX LR**

void f8(int16_t a[], int32_t k)
{
a[k+1] = 0 ;
}
	f8: LDR R2,=0
	ADD R1,R1,1
	ADD R0,R0,R1
	STRH R2,[R0,LSL 1]
	BX LR
		**f8: LDR R2,=0
		LSL R1,R1,1
		ADD R1,R1,2
		STRH R2,[R0,R1]
		BX LR**
			f8: LDR R2,=0
			LSL R1,R1,1
			STRH R2,[R0,R1,2]
			BX LR

int32_t f9(int32_t a32)
{
return a32 % 10 ;
}
	f9: LDR R1,=10
	SDIV R2,R0,R1
	MLS R0,R0,R1,R2
	BX LR
		**f9: LDR R1,=10
		SDIV R2,R0,R1
		MLS R0,R2,R1,R0
		BX LR**
			f9: LDR R1,=10
			SDIV R2,R0,R1
			MLS R0,R0,R2,R1
			BX LR

int64_t f10(int32_t a32)
{
return 100 * a32 ;
}
	f10: LDR R1,=100
	SMULL R0,R1,R0,R1
	BX LR
		**f10: LDR R1,=100
		MUL R0,R0,R1
		ASR R1,R0,31
		BX LR**
			f10: LDR R1,=100
			MUL R0,R0,R1
			LDR R1,=0
			BX LR

int64_t f11(int64_t a64)
{
if (a64 < 0) a64 = 0 ;
return a64 ;
}
	f11: CMP R1,0
	BLT L1
	LDR R0,=0
	LDR R1,=0
	L1: BX LR
		f11: CMP R1,0
		BGT L1
		LDR R0,=0
		LDR R1,=0
		L1: BX LR
			**f11: CMP R1,0
			BGE L1
			LDR R0,=0
			LDR R1,=0
			L1: BX LR**

void f12(int16_t **pp16)
{
((pp16 + 1) + 1) = 0 ;
}
	**f12: LDR R1,=0
	LDR R2,[R0,4]
	STRH R1,[R2,2]
	BX LR**
		f12: LDR R1,=0
		LDR R2,[R0,1]
		STRH R1,[R2,2]
		BX LR
			f12: LDR R1,=0
			LDR R2,[R0,4]
			STRH R1,[R2,1]
			BX LR

Points this page: _____

9. [4 pts ea] In each of the two code fragments below, is the compound conditional a logical **AND** or a logical **OR**? (Circle one of the two words in each comment line.)

```
        LDR   R0,x      // AND or OR ?
        CMP   R0,10
        BLT   L1
        CMP   R0,20
        BLE   L2
L1:     LDR   R0,=0
        STR   R0,x
L2:
```

```
        LDR   R0,x      // AND or OR ?
        CMP   R0,10
        BLT   L2
        CMP   R0,20
        BGT   L2
L1:     LDR   R0,=0
        STR   R0,x
L2:
```

10. [10 pts ea] Translate the following C into ARM assembly:

C Function	*ARM Assembly*

Int64_t Choose(int64_t a64, int64_t b64)
{
return (a64 < b64) ? (b64 + 1) : (b64 – 1) ;
}
	// Use an IT block
	
	Choose: SUBS R0,R0,R2
	SBCS R1,R1,R3
	ITTEE LT
	ADDSLT R0,R2,1
	ADCLT R1,R3,0
	SUBSGE R0,R2,1
	SBCGE R1,R3,0
	BX LR

int32_t Usable(int32_t min, int32_t max)
{
int32_t random(void) ;
int32_t temp ;
 	
temp = random() ;
 	
if (temp >= min && temp <= max)
return temp ;
else
return 0 ;
}
	// Do NOT use IT block
	
	Usable: PUSH {R4,R5,LR}
	MOV R4,R0
	MOV R5,R1
	BL Random
	CMP R0,R4
	BLT Else
	CMP R0,R5
	BLE EndIf
	Else: LDR R0,=0
	EndIf: POP {R4,R5,PC}