NAME: ___________________________________

SANTA CLARA UNIVERSITY
Department of Computer Engineering

COEN 020	Midterm Exam	Fall 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] How many different values can you represent with 8 binary bits?

   a. 64 b. 128 **c. 256** d. 512

2. [4 pts] Given a two's complement number with two integer bits and two fractional bits, what would be the 4-bit representation of a negative number with the smallest magnitude?

   a. 10.01 b. 11.00 **c. 11.11** d. 00.01

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

   $\mathbf{33_{10} = 1x5^2 + 1x5^1 + 3x5^0 \rightarrow 113_5}$

4. [5 pts] Convert the unsigned radix 4 value 231.3_4 to base 16 (hex).

   $\mathbf{231.3_4 = 02\ \ 31\ \ .\ \ 30 \rightarrow 2D.C_{16}}$

5. [5 pts] Convert signed 2's complement value 1100.01_2 to decimal.

   $\mathbf{1100.01_2 = -\ 0011.11_2 = -3.75_{10}}$

6. [5 pts] Convert -36_{10} to an 8-bit 2's complement representation.

   $\mathbf{36 = 32 + 4 = 2^5 + 2^2 \rightarrow 100100_2 \rightarrow 00100100_2}$

   $\mathbf{-00100100_2 \rightarrow 11011100_2}$

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 f0(int32_t a32)
{
return a32 + 5 ;
}
	f0: LDR R0,a32
	ADD R0,R0,5
	BX LR
		f0: ADD R0,R0,5
		BX LR
			f0: ADD R0,R0,5
			STR R0,f0
			BX LR

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

void f3(int32_t a32[ ])
{
a32[5] = 0 ;
}
	f3: LDR R1,=0
	ADR R0,a32
	STR R1,[R0,10]
	BX LR
		f3: LDR R1,=0
		STR R1,[R0,5]
		BX LR
			f3: LDR R1,=0
			STR R1,[R0,20]
			BX LR

void f4(uint64_t *p64)
{
*p64 = 5 ;
}
	f4: LDRD R1,R2,=5
	STRD R1,R2,[R0]
	BX LR
		f4: LDR R1,=5
		LDR R2,=0
		STRD R1,R2,[R0]
		BX LR
			f4: LDR R1,=5
			LDR R2,=0
			STRD R1,R2,[p64]
			BX LR

uint8_t f5(uint8_t u8)
{
if (u8 != 0) --u8 ;
return u8 ;
}
	f5: CMP R0,0
	IT NE
	SUB R0,R0,1
	BX LR
		f5: CMP R0,0
		ITT NE
		SUBNE R0,R0,1
		BX LR
			f5: CMP R0,0
			IT NE
			SUBNE R0,R0,1
			BX LR

void f6(int32_t a32[])
{
int32_t *p32 ;
p32 = a32 + 1 ;
*p32 = 0 ;
}
	f6: ADD R0,R0,1
	LDR R1,=0
	STR R1,[R0]
	BX LR
		f6: LDR R1,=0
		STR R1,[R0,4]
		BX LR
			f6: ADD R0,R0,4
			STR R0,p32
			LDR R1,=0
			STR R1,[R0]
			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.

uint64_t f7(uint64_t a64)
{
return a64 + a64 ;
}
	f7: ADD R0,R0,R0
	ADC R1,R1,R1
	BX LR
		f7: ADDS R0,R0,R0
		ADD R1,R1,R1
		BX LR
			f7: ADDS R0,R0,R0
			ADC R1,R1,R1
			BX LR

void f8(int32_t a[], int32_t k)
{
a[2*k] = 0 ;
}
	f8: LDR R2,=0
	STR R2,[R0,R1,LSL 3]
	BX LR
		f8: LDR R2,=0
		ADD R1,R1,R1
		STR R2,[R0,R1]
		BX LR
			f8: LDR R2,=0
			ADD R0,R0,R1
			STR R2,[R0,R1]
			BX LR

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

int32_t f10(int32_t a)
{
return 5*(7-a) ;
}
	f10: RSB R0,R0,7
	LDR R1,=5
	SMUL R0,R0,R1
	BX LR
	`
		f10: RSB R0,R0,7
		LDR R1,=5
		MULS R0,R0,R1
		BX LR
			f10: RSB R0,R0,7
			LDR R1,=5
			MUL R0,R0,R1
			BX LR

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

void f12(int16_t *p16)
{
*(p16 + 1) += 1 ;
}
	f12: ADD R0,R0,2
	LDRH R1,[R0]
	ADD R1,R1,1
	STRH R1,[R0]
	BX LR
		f12: ADD R0,R0,1
		LDRH R1,[R0]
		ADD R1,R1,1
		STRH R1,[R0]
		BX LR
			f12: LDR R0,p16+1
			ADD [R0],[R0],1
			BX LR

Points this page: ________________

9. [2 pts ea] True/False:

   a. **False** UMULL sets the overflow flag (V) if an overflow occurs during multiplication.

   b. **True** The MUL instruction works for both signed and unsigned multiplication

   c. **False** The DIV instruction works for both signed and unsigned division

   d. **False** The V flag is used to detect overflow during signed and unsigned addition.

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

C Function	***ARM Assembly***

int32_t Pass(uint32_t u32)
{
if (u32 >= 60 && u32 <= 100) return 1 ;
return 0 ;
}
	// Use conditional branch instructions
	Pass: CMP R0,60
	BLO L1
	CMP R0,100
	BHI L1
	LDR R0,=1
	B L2
	L1: LDR R0,=0
	L2: BX LR

int32_t foo(int32_t *a32, int32_t *b32)
{
bar() ;
return (a32 < b32) ? a32 : b32 ;
}
	// Use an IT block
	I decided to throw out this problem and grade the midterms based on 106 points instead of 116 because the parameters were declared incorrectly. As a result, people either ignored the fact that the parameters were specified as pointers, or they coded the function accordingly, but then didn't note that the function was supposed to return an int, when in fact it was returning a pointer. My bad!