Digital Electronics & Computer Engineering (E85)

Harris Spring 2017

Final Exam

This is a closed-book take-home exam. Electronic devices including calculators are not allowed, except on the computer question on the last page. You are permitted two 8.5x11" sheets of paper with notes.

You are bound by the HMC Honor Code while taking this exam.

The exam is intended to be doable in 3 hours if you have prepared adequately. However, there will be no limit on the time you are allowed except that the written portion must be completed in one contiguous block of time and the computer part must be completed in another contiguous block of time. A contiguous block of time is a period of time working at a desk without breaking for meals, naps, socializing, etc. Please manage your time wisely and do not let the exam expand to take more time than is justified.

Return the exam to Sydney Torrey in the Engineering Department Office no later than Friday 5/12 at noon (5/5 at 5 pm for seniors).

Alongside each question, the number of points is written in brackets. The entire exam is wort points. Plan your time accordingly. All work and answers should be written directly on this examination booklet, except for printouts. Use the backs of pages if necessary. Write neatly; illegible answers will be marked wrong. Show your work for partial credit.

Name: _____

Do Not Write Below This Point

Page 2:	_____	/ 6
Page 3:	_____	/ 5
Page 4-6:	_____	/ 9
Page 7-8:	_____	/ 13
Page 9:	_____	/ 3
Page 10-12:	_____	/ 6
Page 13-14:	_____	/ 7
Page 15-18:	_____	/ 8
Page 20:	_____	/ 6
Total:	_____	/ 63

[1] How many bits are in a byte?

Bits: _____

[2] Write the closest approximation you can make to -3.3 as a 2's complement fixed-point number with 4 integer and 4 fractional digits.

Number: _____

[3] Write -3.375 as an IEEE single-precision floating-point number. Express your result in binary.

Number: _____

The HMC alumni association wants to build a system to help alumni decide whether to attend alumni weekend. It should take three inputs (F: friends, V: years since graduation is multiple of five, W: has to work) and produce one output (A: attend). Specifically, it should recommend attending if one has friends to visit. One should also attend if one's years since graduation is a multiple of five and one doesn't have to work that day.

[1] Is this circuit combinational or sequential? Explain.

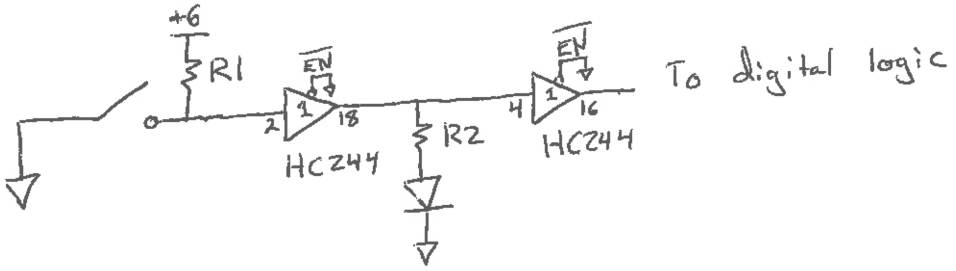
Combinational / Sequential

[1] Write a minimal sum-of-products Boolean equation for A.

Equation: _____

[3] Sketch a circuit using CMOS transistors to compute A. Use no more transistors than necessary.

Consider the following circuit to control an LED with a button. The circuit also buffers the button value to pass it to a digital output. Assume the LED has a forward voltage of 2.0 V when ON. The datasheet for the 74HC244 tristate buffer is given on the next page. The system operates directly off a 6V battery.



[2] Give an expression for the worst-case quiescent power consumption of the system when the button is pressed. Assume the temperature may be anywhere in the range of -40 to 125 C.

Power: _____

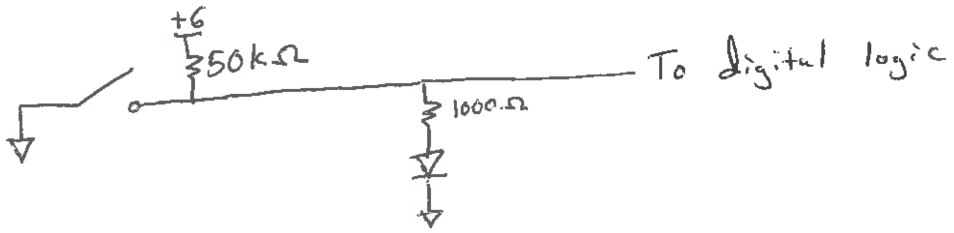
[2] Compute the maximum value of R1.

Max R1: _____

[2] Compute the minimum value of R2 given the attached datasheet.

Min R2: _____

[1] A Caltech engineering intern observes that the buffers perform no logical function and removes them to reduce cost, as shown below. The intern complains to you that the LED never turns ON. Explain why this is observed.



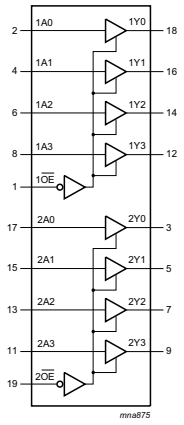
[2] Rewire the intern's circuit so that the switch turns the LED on when pressed. Use no other components.

7. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to GND (ground = 0 V).

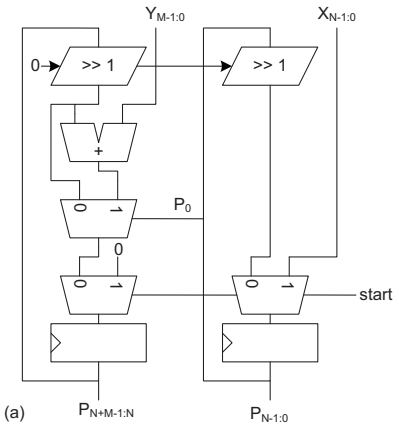
Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	supply voltage		−0.5	+7	V
I_{IK}	input clamping current	$V_I < -0.5$ V or $V_I > V_{CC} + 0.5$ V	-	±20	mA
I_{OK}	output clamping current	$V_O < -0.5$ V or $V_O > V_{CC} + 0.5$ V	-	±20	mA
I_O	output current	-0.5 V $< V_O < V_{CC} + 0.5$ V	-	±35	mA
I_{CC}	supply current		-	70	mA
I_{GND}	ground current		−70	-	mA
T_{stg}	storage temperature		−65	+150	°C
P_{tot}	total power dissipation	SO20, SSOP20, TSSOP20 and DHVQFN20 packages [1]	-	500	mW



Recommended Operating Conditions

Symbol	Parameter	Conditions	25 °C			−40 °C to +85 °C		−40 °C to +125 °C		Unit
			Min	Typ	Max	Min	Max	Min	Max	
74HC244										
V_{IH}	HIGH-level input voltage	V_{CC} = 2.0 V	1.5	1.2	-	1.5	-	1.5	-	V
		V_{CC} = 4.5 V	3.15	2.4	-	3.15	-	3.15	-	V
		V_{CC} = 6.0 V	4.2	3.2	-	4.2	-	4.2	-	V
V_{IL}	LOW-level input voltage	V_{CC} = 2.0 V	-	0.8	0.5	-	0.5	-	0.5	V
		V_{CC} = 4.5 V	-	2.1	1.35	-	1.35	-	1.35	V
		V_{CC} = 6.0 V	-	2.8	1.8	-	1.8	-	1.8	V
V_{OH}	HIGH-level output voltage	$V_I = V_{IH}$ or V_{IL}								
		$I_O = -20$ µA; V_{CC} = 2.0 V	1.9	2.0	-	1.9	-	1.9	-	V
		$I_O = -20$ µA; V_{CC} = 4.5 V	4.4	4.5	-	4.4	-	4.4	-	V
		$I_O = -20$ µA; V_{CC} = 6.0 V	5.9	6.0	-	5.9	-	5.9	-	V
		$I_O = -6.0$ mA; V_{CC} = 4.5 V	3.98	4.32	-	3.84	-	3.7	-	V
		$I_O = -7.8$ mA; V_{CC} = 6.0 V	5.48	5.81	-	5.34	-	5.2	-	V
V_{OL}	LOW-level output voltage	$V_I = V_{IH}$ or V_{IL}								
		I_O = 20 µA; V_{CC} = 2.0 V	-	0	0.1	-	0.1	-	0.1	V
		I_O = 20 µA; V_{CC} = 4.5 V	-	0	0.1	-	0.1	-	0.1	V
		I_O = 20 µA; V_{CC} = 6.0 V	-	0	0.1	-	0.1	-	0.1	V
		I_O = 6.0 mA; V_{CC} = 4.5 V	-	0.15	0.26	-	0.33	-	0.4	V
		I_O = 7.8 mA; V_{CC} = 6.0 V	-	0.16	0.26	-	0.33	-	0.4	V
I_I	input leakage current	$V_I = V_{CC}$ or GND; V_{CC} = 6.0 V	-	-	±0.1	-	±1.0	-	±1.0	µA
I_{OZ}	OFF-state output current	$V_I = V_{IH}$ or V_{IL}; V_{CC} = 6.0 V; $V_O = V_{CC}$ or GND	-	-	±0.5	-	±5.0	-	±10	µA
I_{CC}	supply current	$V_I = V_{CC}$ or GND; I_O = 0 A; V_{CC} = 6.0 V	-	-	8.0	-	80	-	160	µA
C_I	input capacitance		-	3.5	-	-	-	-	-	pF

Consider the following circuit. Assume start is pulsed for one clock cycle and the system runs for N more clock cycles while X and Y are held constant. The arrow on the left side of each right shifter indicates the value shifted into the most significant bit and the arrow on the right side indicates the value shifted out of the least significant bit.



[1] Is this circuit combinational or sequential? Explain.

Combinational / Sequential

[3] Give an expression for the final value in $P_{N+M-1:0}$ as a function of X and Y.

P: _____

Suppose the components have the following delays:

Component	Max / Propagation (ps)	Min / Contamination (ps)
Mux	10	5
Adder	50	12
Shifter (just wires)	0	0
Register clk-to-Q	20	14
Register setup/hold	Setup = 30	Hold = -3

[2] What is the minimum clock period for which the circuit can operate correctly in the absence of clock skew?

Minimum Clock Period: _____

[2] How much clock skew can the system endure before it may become unreliable at any frequency?

Maximum Clock Skew: _____

[5] Write a succinct behavioral (not structural) description of the circuit in Verilog.

```
module mystery #(parameter N = 16, M = 32)
          (input  logic           clk,
           input  logic           start,
           input  logic [N-1:0]   x,
           input  logic [M-1:0]   y,
           output logic [M+N-1:0] p);
```

```
endmodule
```

Consider the following program running on a 32-bit computer. Assume the base address of a is 0x6000.

```
typedef struct auv_id {
     int serialno;
     int manufacturingdate;
     double maxspeed;
     int params[4];
     char name[32];
} auv_id;

int main(void) {
     auv_id a[10];
     double *msptr = &(a[2].maxspeed);
     auv_id *idptr = a + 4;
     char *cptr = &(idptr->name[1]);
}
```

[1] What is the value of `msptr`?

msptr: _____

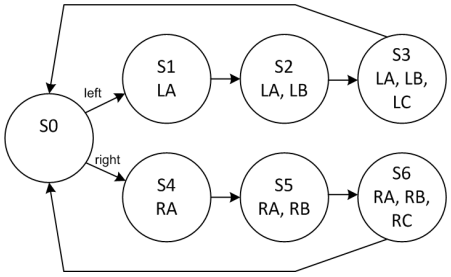
[1] What is the value of `idptr`?

idptr: _____

[1] What is the value of `cptr`?

cptr: _____

Implement the Lab 3 Thunderbird tail lights in C for a Nucleo board. The state transition diagram is given below.

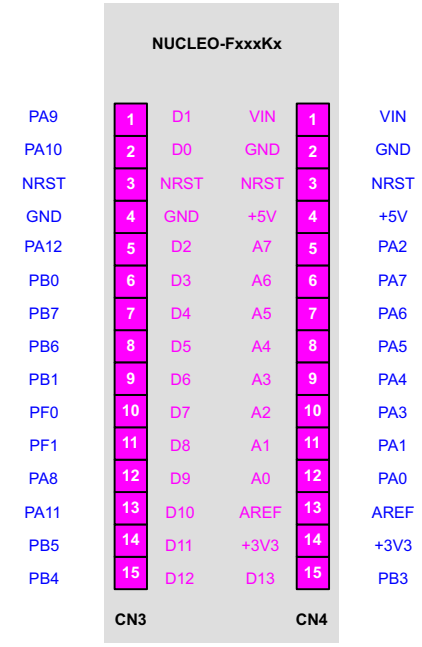


The Nucleo pinout is given below. Assume left and right come from switches connected to D3 and D6 and LA, LB, LC, RA, RB, RC are LEDs driven by A0, A1, A2, A3, A4, and A5, respectively. You may assume that none of the other bits of the ports are connected to anything and that the machine is reset immediately before the program runs. Assume your users have extremely fast eyes so you don't need to introduce delay between states. Do not rely on libraries aside from stm32f042x6.h.

Relevant portions of stm32f042x6.h and the data sheets are given below.

[6] Write your C program.

```
#include <stm32f042x6.h>

int main(void) {









}
```




```
typedef struct
{
  __IO uint32_t MODER;        /*!< GPIO port mode register,                    Address offset: 0x00      */
  __IO uint32_t OTYPER;       /*!< GPIO port output type register,             Address offset: 0x04      */
  __IO uint32_t OSPEEDR;      /*!< GPIO port output speed register,            Address offset: 0x08      */
  __IO uint32_t PUPDR;        /*!< GPIO port pull-up/pull-down register,       Address offset: 0x0C      */
  __IO uint32_t IDR;          /*!< GPIO port input data register,              Address offset: 0x10      */
  __IO uint32_t ODR;          /*!< GPIO port output data register,             Address offset: 0x14      */
  __IO uint32_t BSRR;         /*!< GPIO port bit set/reset register,      Address offset: 0x1A */
  __IO uint32_t LCKR;         /*!< GPIO port configuration lock register,      Address offset: 0x1C      */
  __IO uint32_t AFR[2];       /*!< GPIO alternate function low register,  Address offset: 0x20-0x24 */
  __IO uint32_t BRR;          /*!< GPIO bit reset register,                    Address offset: 0x28      */
} GPIO_TypeDef;

typedef struct
{
  __IO uint32_t CR;            /*!< RCC clock control register,                                  Address offset: 0x00
*/
  __IO uint32_t CFGR;       /*!< RCC clock configuration register,                        Address offset: 0x04 */
  __IO uint32_t CIR;        /*!< RCC clock interrupt register,                            Address offset: 0x08 */
  __IO uint32_t APB2RSTR;   /*!< RCC APB2 peripheral reset register,                      Address offset: 0x0C */
  __IO uint32_t APB1RSTR;   /*!< RCC APB1 peripheral reset register,                      Address offset: 0x10 */
  __IO uint32_t AHBENR;     /*!< RCC AHB peripheral clock register,                       Address offset: 0x14 */
  __IO uint32_t APB2ENR;    /*!< RCC APB2 peripheral clock enable register,               Address offset: 0x18 */
  __IO uint32_t APB1ENR;    /*!< RCC APB1 peripheral clock enable register,               Address offset: 0x1C */
  __IO uint32_t BDCR;       /*!< RCC Backup domain control register,                      Address offset: 0x20 */
  __IO uint32_t CSR;        /*!< RCC clock control & status register,                     Address offset: 0x24 */
  __IO uint32_t AHBRSTR;    /*!< RCC AHB peripheral reset register,                       Address offset: 0x28 */
  __IO uint32_t CFGR2;      /*!< RCC clock configuration register 2,                      Address offset: 0x2C */
  __IO uint32_t CFGR3;      /*!< RCC clock configuration register 3,                      Address offset: 0x30 */
  __IO uint32_t CR2;        /*!< RCC clock control register 2,                            Address offset: 0x34 */
} RCC_TypeDef;

#define GPIOA               ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB               ((GPIO_TypeDef *) GPIOB_BASE)
#define RCC                 ((RCC_TypeDef *) RCC_BASE)
```

Table 19. RCC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLL ON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
	Reset value							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	1
0x04	RCC_CFGR	PLL NODIV	MCOPRE [2:0]			MCO [3:0]				Res.	Res.	PLLMUL[3:0]				PLLXTPRE	PLLSRC[1]	PLLSRC[0]	ADC PRE	Res.	Res.	Res.	PPRE [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
0x08	RCC_CIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	HSI48RDYC	HSI14 RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Res.	HSI48RDYIE	HSI14 RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	HSI48RDYF	HSI14 RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGMCURST	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	ADCRST	Res.	USART8RST	USART7RST	USART6RST	Res.	Res.	Res.	Res.	SYSCFGRST
	Reset value										0				0	0	0		0		0	0		0		0	0	0					0
0x010	RCC_APB1RSTR	Res.	CECRST	DACRST	PWRRST	CRSRST	Res.	CANRST	Res.	USBRST	I2C2RST	I2C1RST	USART5RST	USART4RST	USART3RST	USART2RST	Res.	Res.	SPI2RST	Res.	Res.	WWDGRST	Res.	Res.	TIM14RST	Res.	Res.	TIM7RST	TIM6RST	Res.	Res.	Res.	TIM3RST	TIM2RST
	Reset value		0	0	0	0		0		0	0	0	0	0	0	0			0			0			0			0	0			0	0
0x14	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPFEN	IOPEEN	USART4RST	IOPCEN	IOPBEN	IOPAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	Res.	DAM2EN	DMAEN
	Reset value								0		0	0	0	0	0	0											0		1		1	0	0

Table 24. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = **B..F**)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOx_OTYPER (where x = **A..F**)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOA_OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDR (where x = **B..F**)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = **B..F**)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = **A..F**)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
0x14	GPIOx_ODR (where x = **A..F**)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Consider the following C program

```
int f(void) {
     int i;
     int result = 0;
     for (i=1; i<=5; i++) {
          result = (result + i) & 0xE;
     }
     return result;
}

int g(int a) {
     int q = 0;
     while (q < a) q = (q + f()) << 1;
     return q;
}
```

A literal assembly language translation of `f` is:

```
# result in R4, i in R5
f    push R4, R5    ; save R4 and R5
     mov R4, #0     ; result = 0
     mov R5, #1     ; i = 1
forf cmp R5, #5     ; i <= 5?
     bgt donef      ; no: terminate for
     add R4, R4, R5 ; result = result + i
     and R4, 0x0E   ; result = result & 0xE
     add R5, R5, #1 ; i = i+1
     b forf
donef mov R0, R4    ; return result
     pop R4, R5     ; restore R4 and R2
     mov PC, LR     ; return to caller
```

[3] Literally translate `g` into ARM assembly language without optimization.

[2] What is the execution time of your program computing `f()` on a single-cycle ARM processor operating at 1 GHz?

Execution Time: _____

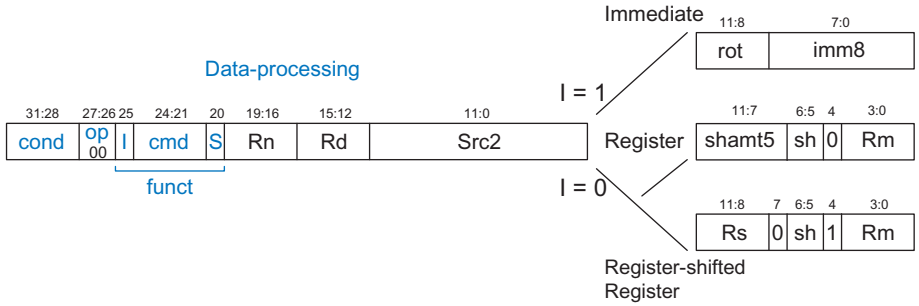
[2] Optimize the program for speed. Write new assembly language code that returns the same answer for `g(a)` but runs as fast as possible. You may make any changes necessary to achieve this goal, not necessarily limited to `g`.

The ARM `MVN Rd, Rn` instruction performs bitwise NOT on Rn and places the result in Rd. It is a data processing instruction with a cmd field of 1111. MVNS does the same, and affects the N and Z flags. Modify the ARM multicycle processor to support the MVN and MVNS instructions, using as little additional hardware as feasible.

[3] Mark up the attached multicycle processor diagram and ALU to handle the new instructions.

[2] Mark up the attached multicycle controller (including state transition diagram and truth tables) to handle the new instructions.

[2] The attached multicycle memfile.s test code has highlighted modifications to test the new instruction. Translate these three new lines of assembly to machine language. Express your code in hexadecimal. The format for a data processing instruction is given below. The cmd field for ORR is 1100 and the cond field for ALWAYS is 1110.



MVN R9, R2: _____

ORR R9, R9, #1: _____

MVN R2, R9: _____

[1] Predict what value should be written to mem[248] at the last line of the program.

Predicted Value: _____

Multicycle Processor

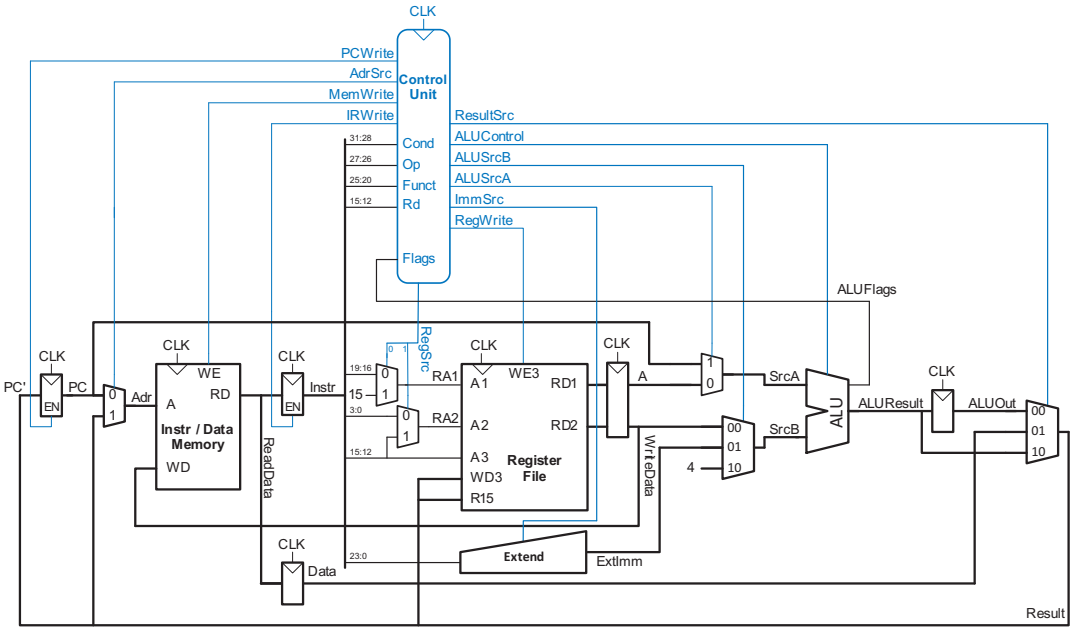
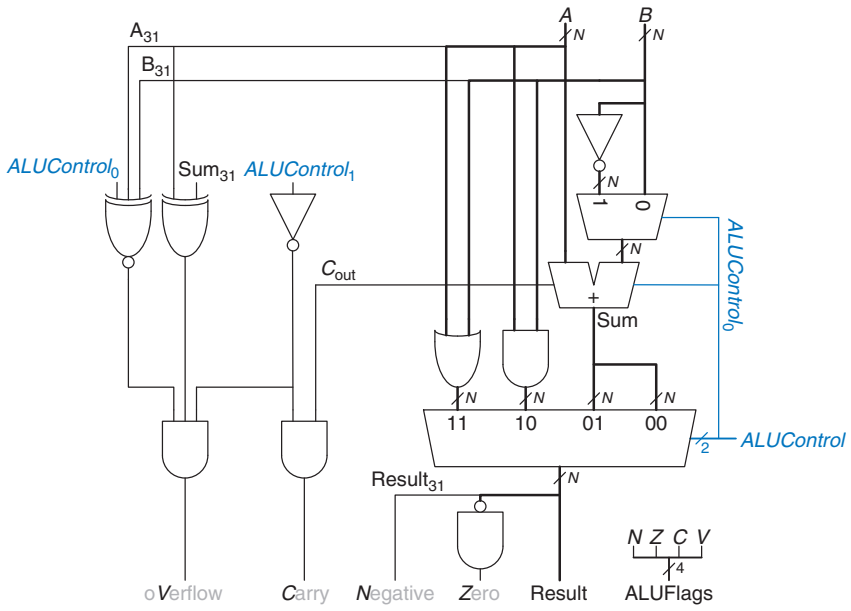


Figure 7.30 Complete multicycle processor

ALU



Multicycle Controller

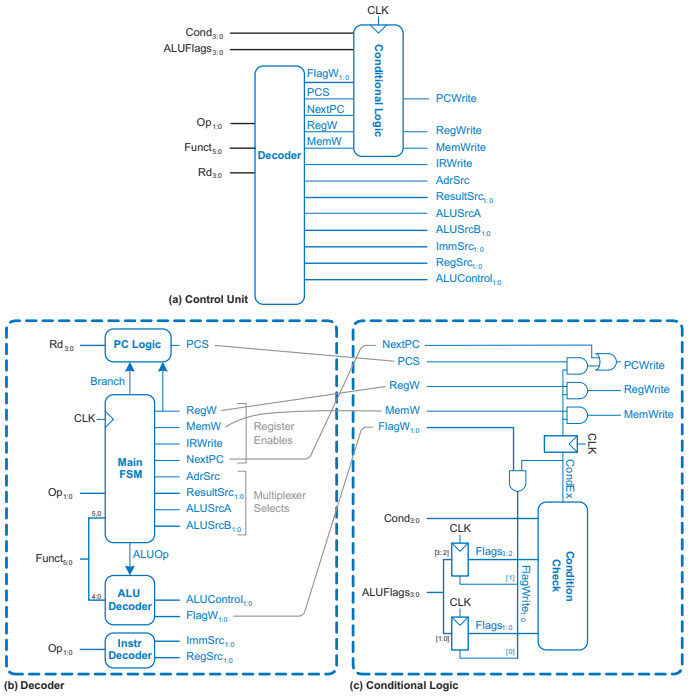


Figure 7.31 Multicycle control unit

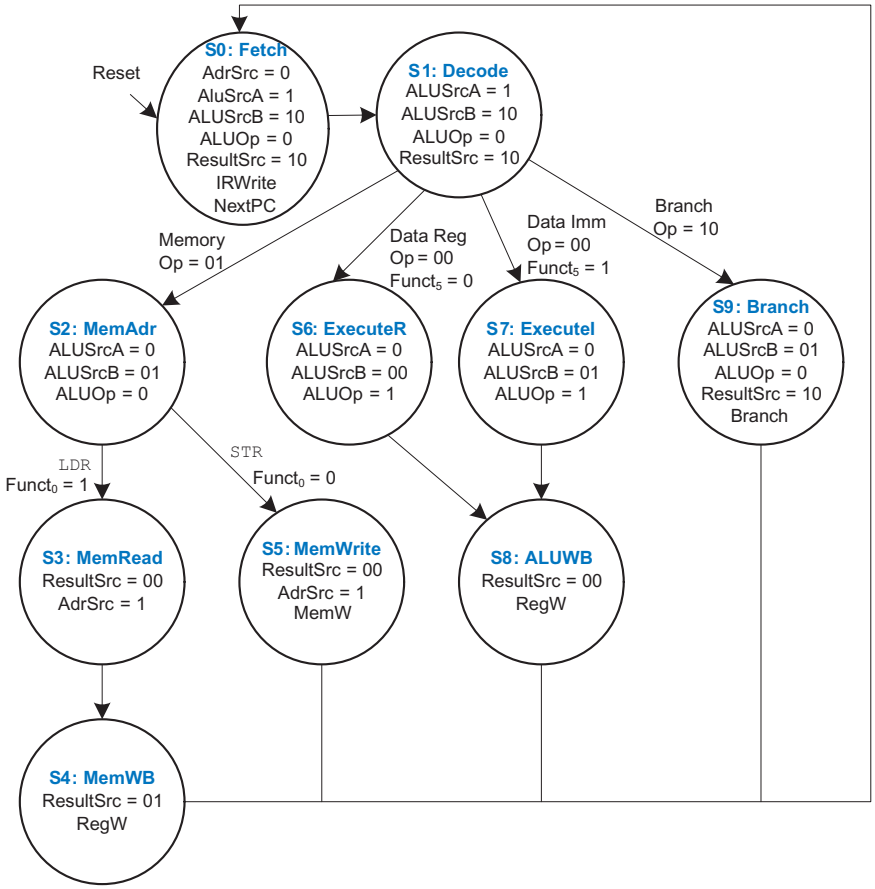


Table 7.6 Instr Decoder logic for *RegSrc* and *ImmSrc*

Instruction	Op	$Funct_5$	$Funct_0$	$RegSrc_1$	$RegSrc_0$	$ImmSrc_{1:0}$
LDR	01	X	1	X	0	01
STR	01	X	0	1	0	01
DP immediate	00	1	X	X	0	00
DP register	00	0	X	0	0	00
B	10	X	X	X	1	10

$ALUOp$	$Funct_{4:1}$ (cmd)	$Funct_0$ (S)	Type	$ALUControl_{1:0}$	$FlagW_{1:0}$
0	X	X	Not DP	00 (Add)	00
1	0100	0	ADD	00 (Add)	00
		1			11
	0010	0	SUB	01 (Sub)	00
		1			11
	0000	0	AND	10 (And)	00
		1			10
	1100	0	ORR	11 (Or)	00
		1			10

```
        ; memfile.dat
MAIN
SUB R0, R15, R15       ; R0 = 0            1110 000 0010 0 1111 0000 0000 0000 1111 E04F000F 0x00
ADD R2, R0, #5         ; R2 = 5            1110 001 0100 0 0000 0010 0000 0000 0101 E2802005 0x04
ADD R3, R0, #12        ; R3 = 12           1110 001 0100 0 0000 0011 0000 0000 1100 E280300C 0x08
SUB R7, R3, #9         ; R7 = 3            1110 001 0010 0 0011 0111 0000 0000 1001 E2437009 0x0c
ORR R4, R7, R2         ; R4 = 3 OR 5 = 7   1110 000 1100 0 0111 0100 0000 0000 0010 E1874002 0x10
AND R5, R3, R4         ; R5 = 12 AND 7 = 4 1110 000 0000 0 0011 0101 0000 0000 0100 E0035004 0x14
ADD R5, R5, R4         ; R5 = 4 + 7 = 11   1110 000 0100 0 0101 0101 0000 0000 0100 E0855004 0x18
SUBS R5, R5, #10       ; R5 = 11 - 10 = 1  1110 001 0010 1 0101 0101 0000 0000 1010 E255500A 0x1c
SUBSGT R5, R5, #2      ; R5 = 1 - 2 = -1   1100 001 0010 1 0101 0101 0000 0000 0010 C2555002 0x20
ADD R5, R5, #12        ; R5 = -1 + 12 = 11 1110 001 0100 0 0101 0101 0000 0000 1100 E285500C 0x24
SUBS R8, R5, R7        ; R8 = 11 - 3 = 8   1110 000 0010 1 0101 1000 0000 0000 0111 E0558007 0x28
BEQ END                ; not taken         0000 1010 0000  0000 0000 0000 0000 1100 0A00000F 0x2c
SUBS R8, R3, R4        ; R8 = 12 - 7  = 5  1110 000 0010 1 0011 1000 0000 0000 0100 E0538004 0x30
BGE AROUND             ; should be taken   1010 1010 0000  0000 0000 0000 0000 0000 AA000000 0x34
ADD R5, R0, #0         ; should be skipped 1110 001 0100 0 0000 0101 0000 0000 0000 E2805000 0x38
AROUND
SUBS R8, R7, R2        ; R8 = 3 - 5 = -2   1110 000 0010 1 0111 1000 0000 0000 0010 E0578002 0x3c
ADDLT R7, R5, #1       ; R7 = 11 + 1 = 12  1011 001 0100 0 0101 0111 0000 0000 0001 B2857001 0x40
SUB R7, R7, R2         ; R7 = 12 - 5 = 7   1110 000 0010 0 0111 0111 0000 0000 0010 E0477002 0x44
STR R7, [R3, #224]     ; mem[12+224] = 7    1110 010 1100 0 0011 0111 0000 0101 0100 E58370E0 0x48
LDR R2, [R0, #236]     ; R2 = mem[236] = 7  1110 010 1100 1 0000 0010 0000 0110 0000 E59020EC 0x4c
ADD R15, R15, R0       ; PC <- PC + 8      1110 000 0100 0 1111 1111 0000 0000 0000 E08FF000 0x50
ADD R2, R0, #14        ; shouldn't happen  1110 001 0100 0 0000 0010 0000 0000 0001 E280200E 0x54
MVN R9, R2
ORR R9, R9, #1
MVN R2, R9
B END                  ; always taken      1110 1010 0000  0000 0000 0000 0000 0001 EA000001 0x64
ADD R2, R0, #13        ; shouldn't happen  1110 001 0100 0 0000 0010 0000 0000 0001 E280200D 0x68
ADD R2, R0, #10        ; shouldn't happen  1110 001 0100 0 0000 0010 0000 0000 0001 E280200A 0x6C
END
STR R2, [R0, #248]     ; mem[248] = ?      1110 010 1100 0 0000 0010 0000 1111 1000 E58020F8 0x70
```

END OF WRITTEN PORTION OF EXAM

DO NOT PROCEED PAST THIS POINT UNTIL YOU ARE PREPARED TO CEASE ALL WORK ON THE WRITTEN PORTION AND MOVE ON TO THE COMPUTER PORTION.

COMPUTER PORTION OF EXAM

Once you start this question, you may refer to the written portion of the exam, but may not spend any more time on the written portion or change any of your answers on that portion.

Modify your ARM multicycle processor from Lab 11 to support the MVN instruction. Modify your memfile.dat to add the three new lines of machine language code from the previous question. Simulate your modified code.

[2] Print out your Verilog code and circle or highlight the lines you modified.

[4] Print out a simulation waveform showing at least the value being written to memory location 248 on the last cycle. Circle this value in the waveform.