

Name _____

CSE 241

Final Exam

Dec 21 2009

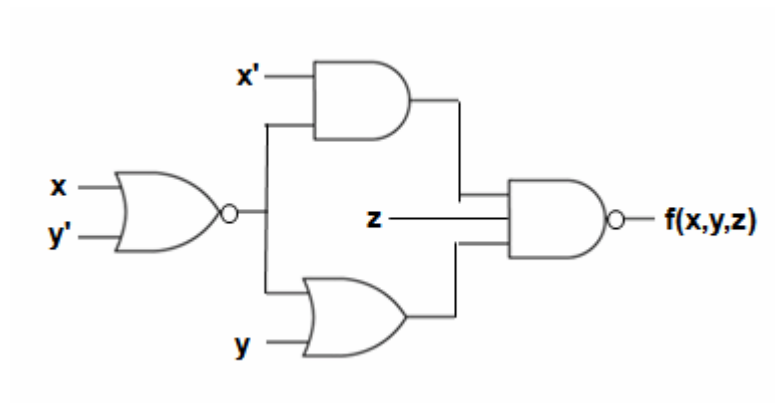
Instructions: 3 hours closed book, notes. Place answers in answer boxes at the bottom of each page. Show all work in blank space above the answer box. Do not burst staple. No electronic devices permitted.

(a) Let X be the base-16 number 1A. Write the number $-X$ in base-8 7's complement form. Your answer should have 4 base-8 digits, ie. use the offset origin which is displaced from the true origin by $10^4_{(8)}$.

(b) Let $Z=Y/X$ where $X=14$, $Y=294577$ and all 3 numbers are base-16. Find Z rounded to the nearest integer.

1 (a)	1 (b)
-------	-------

2.



2(a) Express $f(x,y,z)$ as a DNF Boolean function.

2(b) Express $f(x,y,z)$ in minterm canonical form.

2(a)	2(b)
------	------

3.

$$f(w, x, y, z) = x + (x'yz' + z)' + (w + y + z)'$$

(a) Draw any circuit for $f(w, x, y, z)$ which uses only OR and AND gates. No bubbles or inverting amplifiers are allowed. Assume double-rail logic.

(b) Convert your circuit from OR and AND gates to one using just NAND gates. Draw the resulting circuit.

3 (a)

3 (b)

4.

$$f(w, x, y, z) = \sum m(0, 1, 2, 8, 9, 10)$$

(a) Sketch the Karnaugh Map for f . Show the maximal subcubes.

(b) Find the minimal DNF realization for f .

(c) Find the minimal CNF realization for f .

4 (a)

yz

wx

4 (b)

4 (c)

5. Answer each part of this question. The parts are not related.

(a) Suppose $f(x,y,z)$ has a prime implicate $g(x,y,z)$. Does f imply g , or vice-versa? Illustrate your answer by giving any specific $f(x,y,z)$, identifying one of its prime implicates, and showing that your answer works in this example case.

(b) Suppose each non-empty column of the Quine-McCluskey algorithm for a given f ends up with exactly one entry without a check mark (one unmatched entry). What does that tell us about its prime implicants?

(c) Is every 1-cell in a Karnaugh Map necessarily covered by at least one essential prime implicant? If yes explain, if no give an example.

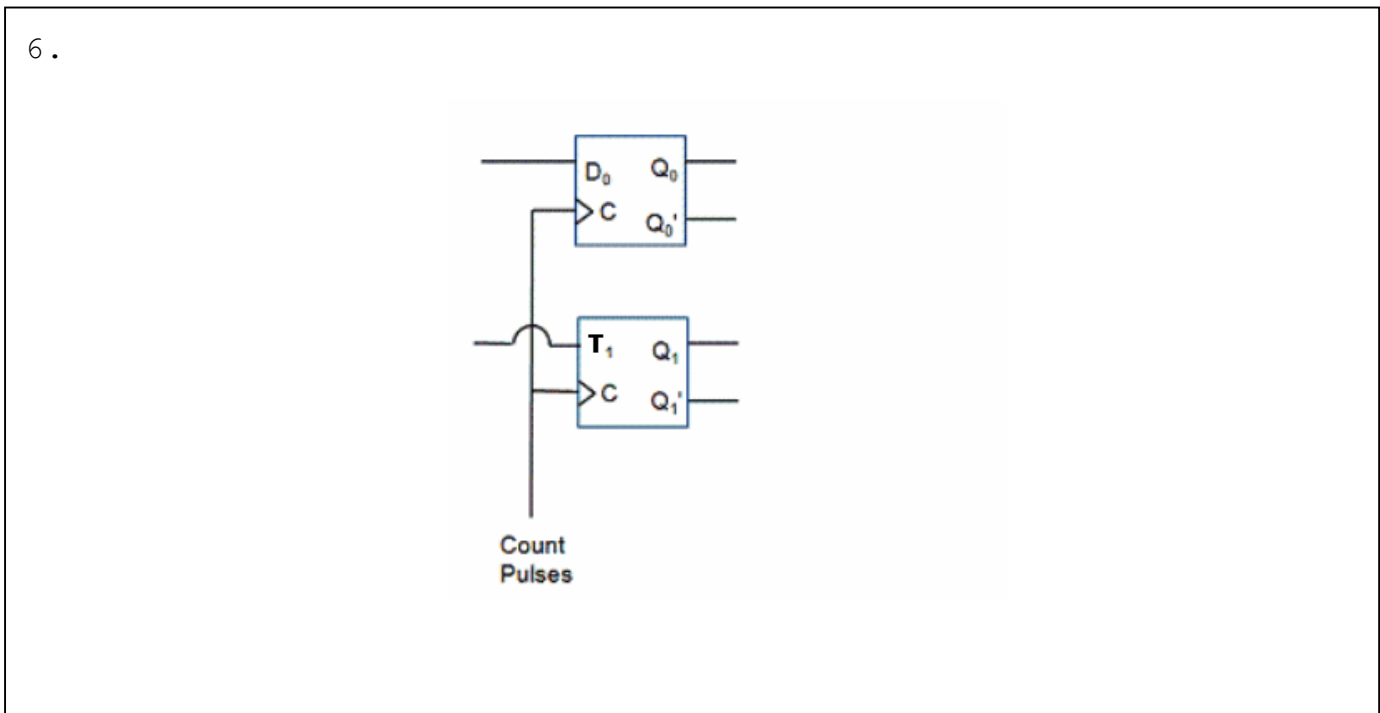
6. Design a circuit which produces the repeating output sequence shown in the table to the left. Starting from $Q_0Q_1=01$, after the next positive clock edge the outputs go to 10, then 00, then back to 01, and so on. Use a D flip-flop for Q_0 and T flip-flop for Q_1 (both positive edge-triggered) as shown in the answer box. The function tables for both types of flip-flop are shown to the right. Complete the circuit diagram begun in the answer box.

Flip-flop function tables

Q_0	Q_1
0	1
1	0
0	0

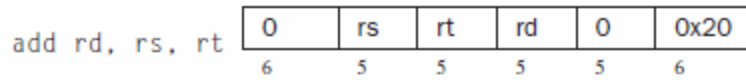
Inputs		Outputs	
D	C	Q^+	Q'^+
0	↑	0	1
1	↑	1	0
X	0	Q	Q'

Inputs		Outputs	
T	C	Q^+	Q'^+
0	↑	Q	Q'
1	↑	Q'	Q
X	0	Q	Q'

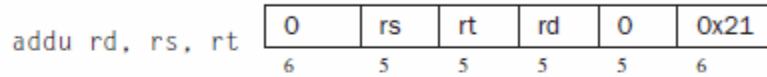


Reference Sheet

Addition (with overflow)

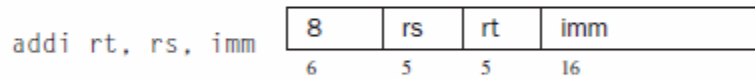


Addition (without overflow)



Put the sum of registers *rs* and *rt* into register *rd*.

Addition immediate (with overflow)

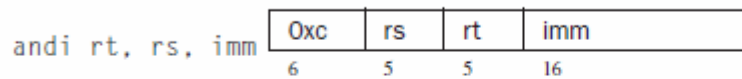


Addition immediate (without overflow)



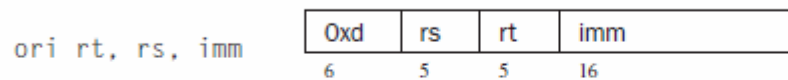
Put the sum of register *rs* and the sign-extended immediate into register *rt*.

AND immediate

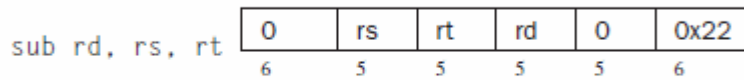
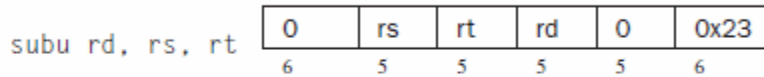


Put the logical AND of register *rs* and the zero-extended immediate into register *rt*.

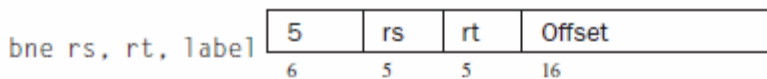
OR immediate



Put the logical OR of register *rs* and the zero-extended immediate into register *rt*.

Subtract (with overflow)**Subtract (without overflow)**

Put the difference of registers *rs* and *rt* into register *rd*.

Branch on not equal

Conditionally branch the number of instructions specified by the offset if register *rs* is not equal to *rt*.

Branch on equal zero

beqz *rsrc*, label *pseudoinstruction*

Conditionally branch to the instruction at the label if *rsrc* equals 0.

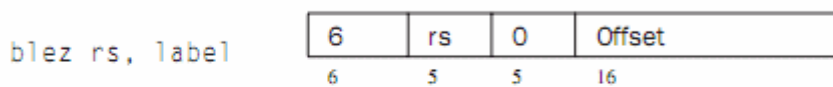
Branch on greater than equal

bge *rsrc1*, *rsrc2*, label *pseudoinstruction*

Branch on greater than equal unsigned

bgeu *rsrc1*, *rsrc2*, label *pseudoinstruction*

Conditionally branch to the instruction at the label if register *rsrc1* is greater than or equal to *rsrc2*.

Branch on less than equal zero

Conditionally branch the number of instructions specified by the offset if register *rs* is less than or equal to 0.

Jump



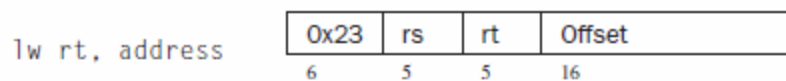
Unconditionally jump to the instruction at target.

Load immediate

li rdest, imm *pseudoinstruction*

Move the immediate *imm* into register *rdest*.

Load word



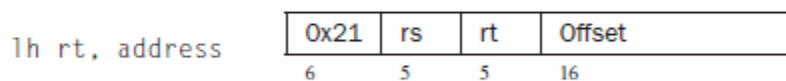
Load the 32-bit quantity (word) at *address* into register *rt*.

Load address

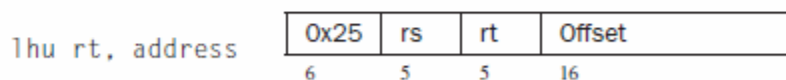
la rdest, address *pseudoinstruction*

Load computed *address*—not the contents of the location—into register *rdest*.

Load halfword

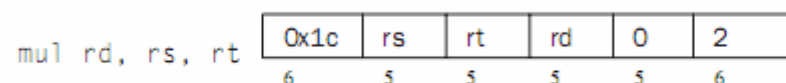


Load unsigned halfword



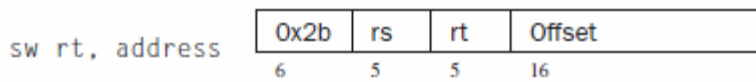
Load the 16-bit quantity (halfword) at *address* into register *rt*. The halfword is sign-extended by *lh*, but not by *lhu*.

Multiply (without overflow)



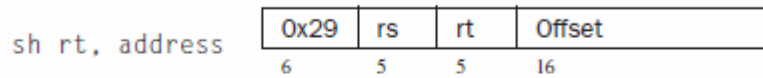
Put the low-order 32 bits of the product of *rs* and *rt* into register *rd*.

Store word



Store the word from register *rt* at *address*.

Store halfword



Store the low halfword from register *rt* at *address*.

Syscalls

Service	System call code	Arguments	Result
print_int	1	\$a0 = integer	
print_float	2	\$f12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer (in \$v0)
read_float	6		float (in \$f0)
read_double	7		double (in \$f0)
read_string	8	\$a0 = buffer, \$a1 = length	
sbrk	9	\$a0 = amount	address (in \$v0)
exit	10		
print_char	11	\$a0 = char	
read_char	12		char (in \$a0)
open	13	\$a0 = filename (string), \$a1 = flags, \$a2 = mode	file descriptor (in \$a0)
read	14	\$a0 = file descriptor, \$a1 = buffer, \$a2 = length	num chars read (in \$a0)
write	15	\$a0 = file descriptor, \$a1 = buffer, \$a2 = length	num chars written (in \$a0)
close	16	\$a0 = file descriptor	
exit2	17	\$a0 = result	

FIGURE A.9.1 System services.