Exam 1 - 1 EECE 344 Fall 2008

Name: Mariano Valdez

Student Number: 100818865

Problem	Possible	Score
1	15	15
2	20	15
3	20	19
4	15	12
5	20	14
6	15	10
Total	105	85

1. General information question:
   a) What is the basic tenet of all stored program computers?

Fetch → Decode → Execute

   b) Identify the four different types of instructions and give an example of each from the PowerPC instruction set.

Work: add — Movement: stw — Program Control: bl — System Control: mtcr

   c) When a non-critical interrupt occurs, where is the return address for the system stored?

SRR0

   d) We have used a mnemonic instruction `lis` to load the upper 16 bits of a register. What is the instruction that is actually invoked in order to do this work?

addis rD, 0, 0x0000

   e) Assume that the 4 LEDs at the edge of the trainer board have been associated with the base address of 0x81410000. What value at what address is used to make sure the pins associated with the 4 LEDs are established as outputs?

0x0 → 0x81410004

   f) Assume that register 9 contains 0x80001080. What address is accessed by the instruction '`lwz r8, 0x40(r9)`'?

0x 80001080

   g) The PowerPC, like most other processors, has a dual mode of operation (privileged mode, user mode). What is the purpose of a dual mode system?

there is more security in privileged mode
when exceptions occur processor shift to privileged mode

2. Subroutine question: A programmer wrote a small subroutine to wait for the PIS bit in the Timer-Status Register to be set; then to clear the bit, write 1's to the 4 LEDs (located at address 0x81400000) and return. This subroutine was used in a system that needed a timer, but without using the interrupt system. The programmer called this routine from a larger system handling routine. This code fragment is as follows:

```
Addr   Bits              Instr
5050 60000000            …
5054 480000AD            bl subrtn
5058 60000000            …

5100 3C808140  subrtn:   lis r4,0x8140
5104 7D18F2A6  again:    mfspr r8,0x3d8      # addr/number of TSR
5108 750B0800            andis. r11,r8,0x0800
510c 41820008            bt 2,goback
5110 4BFFFFF4            b again
5114 3CE00800  goback:   lis r7,0x0800
5118 7CF8F3A6            mtspr 0x3d8,r7
511c 39A0000F            li r13,15
5120 91A40000            stw r13,0(r4)
5124 4E800020            blr
```

r4 = 0x81400000
r8 = 0x08000000
r11 = 0x08000000
r7 = 0x08000000
r13 = 0x0000000F

B ⇒ 1011
9 1001
1001

This question deals with the registers used in the routine. Below is a before and after representation for 16 of the registers. The before values are given (values of registers before executing the instruction at 0x5054); fill in the after values (values of registers after executing the instruction at 0x5054)–like what you expect to happen if you put a breakpoint instruction at 0x5058. The PIT system is enabled and configured to reload from modulus latch, but cause no interrupts. Only mark in the *After* area those registers that have been changed by the above code fragment, and in those boxes place the correct value for the register.

Before		After	
r0 = 0x00000000	r1 = 0x11111111	r0 =	r1 =
r2 = 0x22222222	r3 = 0x33333333	r2 =	r3 =
r4 = 0x44444444	r5 = 0x55555555	r4 = 0x8140 0000	r5 =
r6 = 0x66666666	r7 = 0x77777777	r6 =	r7 = 0x0800 0000
r8 = 0x88888888	r9 = 0x99999999	r8 = 0x9000 0000	r9 =
r10 = 0xAAAAAAAA	r11 = 0xBBBBBBBB	r10 =	r11 = 0x0800 0000
r12 = 0xCCCCCCCC	r13 = 0xDDDDDDDD	r12 =	r13 = 0x0000 000F
r14 = 0xEEEEEEEE	r15 = 0xFFFFFFFF	r14 =	r15 =
lr = 0xABCDEF00	ctr = 0x00000000	lr = 0x00005058	ctr =
cr = 0xFFFFFFFF	tsr = 0x90000000	cr =	tsr =

3. Data movement question: The first laboratory explored moving information to and from memory with the load and store instructions. Below is a small code fragment, followed by a memory and register contents description. Identify the locations in memory and the registers that are changed by the code fragment, and give the updated values.

```
Addr BitPattern      Insruction
2044 3D400001        lis r10,0x0001
2048 614A4560        ori r10,r10,0x4560
204c 816A0014        lwz r11,0x14(r10)
2050 A1CA000A        lhz r14,0x0A(r10)
2054 89EA0017        lbz r15,0x17(r10)
2058 992A0033        stb r9,0x33(r10)
205c B0EA0026        sth r7,0x26(r10)
2060 90AA0038        stw r5,0x38(r10)
2064 60000000
```

r10 = 0x00014560
r11 = CCDDEEFF
r14 = 0x00002233
r15 = 0x000000FF

Before		After	
r0 = 0x00000000	r1 = 0x11111111	r0 =	r1 =
r2 = 0x22222222	r3 = 0x33333333	r2 =	r3 =
r4 = 0x44444444	r5 = 0x55555555	r4 =	r5 =
r6 = 0x66666666	r7 = 0x77777777	r6 =	r7 =
r8 = 0x88888888	r9 = 0x99999999	r8 =	r9 =
r10 = 0xAAAAAAAA	r11 = 0xBBBBBBBB	r10 = 0x00014560	r11 = CCDDEEFF
r12 = 0xCCCCCCCC	r13 = 0xDDDDDDDD	r12 =	r13 =
r14 = 0xEEEEEEEE	r15 = 0xFFFFFFFF	r14 = 0x00002233	r15 = 0x000000FF
LR = 0x00000000	CTR = 0x00000000	LR =	CTR =

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00014560	01	23	45	56	89	AB	CD	EF	00	11	22	33	44	55	66	77
00014570	88	99	AA	BB	CC	DD	EE	FF	12	13	14	15	16	17	18	19
00014580							77	77								
00014590				99					AA	AA	AA	AA				

0023

4. Coding question: In the space provided below, write a code fragment that will create a loop (use the counter register to implement the loop) that will start at address 0x00030400 and fill each word location with its addresses. Do this for 10000 locations.

```
lis r2, 0x0003040@h
ori r2,r2 0x00030400@l  # set up pointer
mtctr 0x3E8 or 1000     # 1000 in decimal
loop: stw r2,0(r2)      # store address of mem in that location
addi r2,r2 (1)4         # increment address in memory / data
bnz loop                # branch until counter = 0
bdnz
```

5. Interrupt question: This question has two parts, and deals only with the UART. The first part is setup/initialization, the second is steady state. In the space provided below, give instructions that will set up the 16550 UART to have a baud rate of 19,200 (pattern for divisor is 0x0145), transmit interrupt not enabled, receive interrupt enabled, no parity, *7 data bits*, and one stop bit. Since this question is about the UART only, don't worry about the interrupt controller.

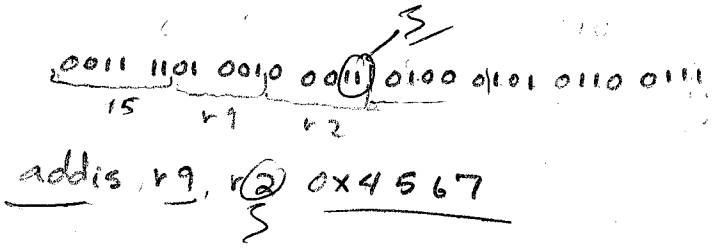
```
.set LCR  0x100C      .set BASEADDR 0x
.set RBR  0x1000                          → IER get path
.set DLL  0x1000                            0x40
.set DLM  0x1004
lis r2, BASEADDR@h
ori r2, r2 BASEADDR@l
li r3, 0x80          # pattern for bit 7 of LCR
sth r3, LCR(r2)      # set Divisor Latch bit in LCR
li r4, 0x01          # divisor for DLL
sth r4, DLM(r2)      # store in DLM (MSB)
li r4, 0x45                                  ? wtf
sth r4, DLL(r2)      # store in DLL
li r4, 0x40          # pattern for 7 data bits, one stop, no parity
sth r4, LCR(r2)      # send to LCR
```

For the second part of this question, give code for an interrupt service routine that will echo any received character and reset appropriate UART flags. Again, don't worry about the interrupt controller.

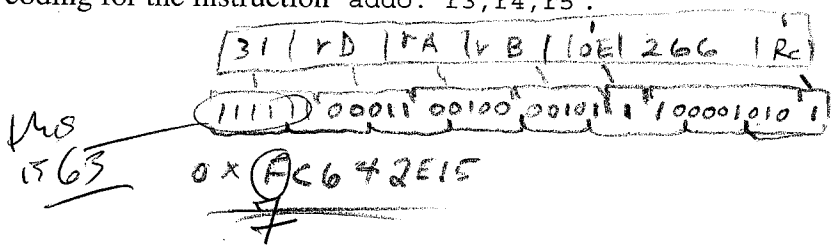
```
Wait: compare LSR (TEMT) bit
      if transmiter is not empty stay in loop
      b Wait

stw r5, 0x1000 (r2)  # send char out to
```

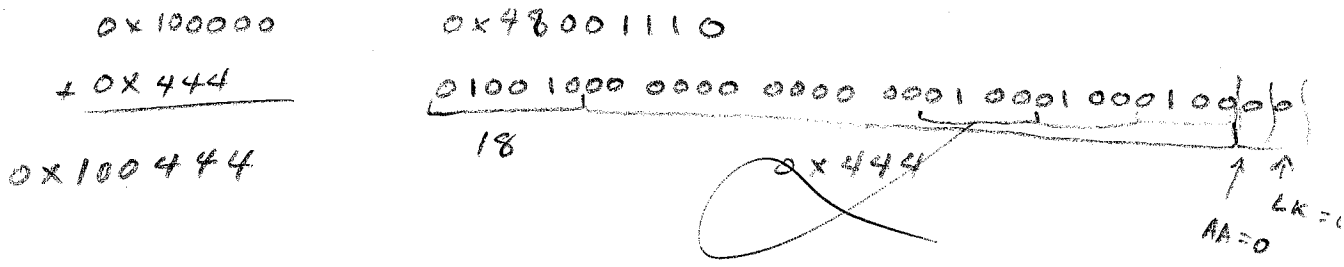
6. Instruction coding question:
   a) What is the instruction represented by the hexadecimal value 0x3d234567.



   b) Give the coding for the instruction 'addo. r3,r4,r5'.



   c) Assume that a branch instruction is located at 0x10000 with the bit pattern 0x48001110. What address is the target of this branch?



b