

1. Information representation question: The PowerPC architecture does work with 32 bit numbers, sometimes in unsigned binary, sometimes in two's complement. It is possible to assume that there is a radix point somewhere in the word, which enables us to consider fractional values as well as whole values. The architecture can also move 16 bit and 8 bit quantities. Complete the table below with the appropriate information – dealing with 16 bit numbers, sometimes as integers, sometimes as fixed point values. Remember that the ‘p’ quantity – in fixed point numbers – identifies the number of places to move the radix point from its normal position at the right of the value.

Number	Bit Pattern	Value of Representation
Value of this bit pattern; 16 bit two's complement integer	1001101010010100 0110010101101011 +	- 25964 ✓
Generate bit pattern for: in 16 bit two's complement integer	00 100 101 001 110	14,000
Value of this bit pattern: 16 bit two's complement, fixed point, p=8	1000100110010110	- 30314 ✓
Generate bit pattern for: In 16 bit two's complement, fixed point, p=8	110 101 011 001 110	42 $\frac{23}{32}$
Bit pattern for biggest possible number, fixed point, two's complement, p=8	0111111111111111	Biggest
Bit pattern for smallest possible number, fixed point, two's complement, p=8	1.000000000000000	Smallest (leftmost on number line)

0110010101101011  
1+

0110010101101010  
14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 = 25964

42  $\frac{23}{32}$  = 42.71875

00101010.11000101  
- 11010101.00111010

11010101.00111001

14,0000 = 11011010110000  
= 00100101001111  
= 00100101001110

1000100110010110  
0111011001101001  
1+

0111011001101010  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 = -30314

## 2. General information question:

- a) What is the basic fundamental principle on which all stored program computers are based?

fetch decode execute

- b) How is the concept of Memory Mapped I/O demonstrated by the PowerPC architecture?

the MMIO look like memory so we can write to them. we can read from them from the memory location they are stored in

- c) What instructions in the PowerPC architecture move multiple register values to and from memory?

LW

SW

- d) George is building a vector table for use with the PowerPC. His table starts at location 0x33330000. An Interrupt Service Routine (ISR) has been created to handle both UART interactions and button pushes. This is a fairly large routine, more than 300 instructions. It starts at location 0x33338080. What instruction should the user place at what address to use this ISR when either a button is pushed or the UART needs service?



0x33330000  
0x00000000  
0x33330100

branch-to-0x33338080 @ 0x33330500

-2

- e) What is the purpose of the CTR register?

This is the counter register is used for several things we assign it to

- f) What are the instruction types? Identify each type and give an example from the PowerPC architecture.

work - add

movement - LW

program control - Branch

System control - Interrupts

- g) Assume that a user has hooked a data source to a bunch of wires controlled by a GPIO IP module, and that the address associated with the module is 0x81480000. Also assume that R4 is set to zero, and that R5 is set to 0x81480000. Give a three instruction sequence that will read into R6 the value on the set of wires.

LIS R4, 0x00000000  
LIS R5, GIE(R1)  
STW R5, 0x81480000  
STW R5, 0x81480000  
STW R5, IPEL(R1)



LIS, 0(R4)

- h) Suppose that the processor clock was set to 268,435,456 hertz. How often would TBU increment?

Every 268,435,456 microseconds.

Time Base Upper

-2



3. Instruction execution question 1: The following routine has created by a user for some strange manipulations on data. Hence, there appears to the observer some un-explainable operations (in other words, don't try to make sense out of what this routine does...) The routine "CTIO3" is called in the code below, and the purpose of this question is to determine the final value of the registers *after* the code fragment executes. When the program activity is at the location indicated "before", a snapshot of the registers is given in the "Before" area of the table. Please indicate in the "After" area of the table the register contents when "after" is reached after execution of the subroutine (remember to enter values only for those registers that have changed).

Addr	Bits	Instruction	
3300	60000000	before: nop	10
3304	4800002D	bl CTIO3	14
3308	60000000	after: nop	
3330	39401234	CTIO3: li r10, 0x1234	
3334	614A5678	ori r10, r10, 0x5678	
3338	39C00004	li r14, 0x04	
333C	7DC903A6	mtctr r14	
3340	7C6D8670	over: srawi r13, r3, 0x10 # shift right arith immed	
3344	4200FFFC	bdnz over	
3348	7082F0F0	xyza: andi. r2, r4, 0xf0f0	
334C	4182FFFC	bt 2, xyza	
3350	7CC74B78	or r7, r6, r9	
3354	7D0F6038	and r15, r8, r12	
3358	61698281	ori r9, r11, 0x8281	
335C	4E800020	blr	

$r10 = 12345678$   
 $r14 = 0xEEEEEEF2$   
 $CTR = 0xEEEEEF2$   
 $r13 = 0xFFFF3333$   
 $r2 = 0x44444040$   
 $r7 = 0xFFFFFFFF$   
 $r15 = 0x88888888$   
 $r9 = 0xBBB BBBB$

Before		After	
$r0 = 0x00000000$	$r1 = 0x11111111$	$r0 = \cancel{(-1)}$	$r1 =$
$r2 = 0x22222222$	$r3 = 0x33333333$	$r2 = \cancel{0x44444040}$	$r3 =$
$r4 = 0x44444444$	$r5 = 0x55555555$	$r4 =$	$r5 =$
$r6 = 0x66666666$	$r7 = 0x77777777$	$r6 =$	$r7 = 0xFFFFFFFF$
$r8 = 0x88888888$	$r9 = 0x99999999$	$r8 =$	$r9 = 0xB BBB BBBB$
$r10 = 0xAAAAAAA$	$r11 = 0xB BBB BBBB$	$r10 = \cancel{0x12345678}$	$r11 = \cancel{(-1)}$
$r12 = 0xC CCCCCCCC$	$r13 = 0xDDDDDDDD$	$r12 =$	$r13 = \cancel{0xFFFF3333}$
$r14 = 0xEEEEEEE$	$r15 = 0xFFFFFFFF$	$r14 = \cancel{0xEEEEEF2}$	$r15 = \cancel{0x88888888}$
$LR \rightarrow$	$0x00000000$	$LR \rightarrow$	$\cancel{3330} (-2)$
$CTR \rightarrow$	$0x00000000$	$CTR \rightarrow$	$0xE EEEEF2$

-3  
-4  
-7

$r14$   
 $10 = 16$   
shifted 16 bits

$r14, 0x04$

$r4 = 0100\ 0100\ 0100\ 0100$   
and  
 $\underline{\quad 1111\ 0000\ 1111\ 0000}$   
 $\quad 0100\ 0000\ 0100\ 0000$

$0x EEEEEE$   
4

$0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011$

$FF\ FF\ FF\ FF\ 3333$

$\begin{array}{r} 1110 \\ + 0100 \\ \hline 101101110010 \end{array}$

$EEE F 2$

$\begin{array}{r} r6 \\ \text{or} \\ r9 \\ \hline 10011001 \end{array}$

$\begin{array}{r} r11 \\ \text{or} \\ 10000010 \end{array}$

$\begin{array}{r} r8 \\ \text{and} \\ r12 \\ \hline 10001000 \end{array}$

$\begin{array}{r} r9 \\ \hline 10001000 \end{array}$

4. Instruction execution question 2: Consider the code segment below. As in the previous question, this question basically is to determine the work done by the instructions, and to modify the registers and the memory segment shown below to reflect the instructions shown. And, once again, don't try to attach any meaning to the work done. Identify in the register area and the memory area only those values that are changed!

Addr	Bits	
11100	3DC00007	before: lis r14, 0x4567
11104	61CE3040	ori r14, r14, 0x3040
11108	89EE0019	lbz r15, 0x19(r14)
1110c	7C855214	add r4, r5, r10
11110	7D495B78	or r9, r10, r11
11114	806E0008	<del>lwz r3, 0x08(r14)</del>
11118	912E002C	<del>stw r9, 0x2C(r14)</del>
1111c	990E0032	<del>stb r8, 0x32(r14)</del>
11120	A1AE000A	<del>lha r14, 0x0A(r14)</del>
11124	B1AE003A	sth r12, 0x3A(r14)

Before		After	
r0 = 0x00000000	r1 = 0x11111111	r0 =	r1 =
r2 = 0x22222222	r3 = 0x33333333	r2 = <del>0x45673040</del>	r3 = 0x99AABBC
r4 = 0x44444444	r5 = 0x55555555	r4 = 0xFFFFFFF	r5 =
r6 = 0x66666666	r7 = 0x77777777	r6 =	r7 =
r8 = 0x88888888	r9 = 0x99999999	r8 =	r9 = 0xB BBB BBB
r10 = 0xAAAAAAA	r11 = 0xB BBB BBB	r10 =	r11 =
r12 = 0xCCCCCCC	r13 = 0xDDDDDDDD	r12 =	r13 = 0xB CC
r14 = 0xEEEEEEEE	r15 = 0xFFFFFFFF	r14 = 0x45673040	r15 = 0xBA

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
45673040	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF	01
45673050	23	45	67	89	AB	CD	EF	FE	DC	BA	98	76	54	32	10	65
45673060													BB	BB	BB	BB
45673070			88								CC	CC				

$$r14 = 0x45673040$$

18 +

~~3040  
3A  
7A~~

~~3040  
32  
30A2~~

~~3040  
0A  
304A~~

$$\begin{array}{r}
 & \text{+ } r5 \\
 & \text{+ } r10 \\
 & \text{--- } r4 \\
 & \text{+ } r10 \\
 & \text{+ } r11 \\
 & \text{--- } r9 \\
 & \text{+ } r10 \\
 & \text{+ } r10 \\
 & \text{--- } r11 \\
 & \text{+ } r10 \\
 & \text{+ } r10 \\
 & \text{--- } r11
 \end{array}$$

$$\begin{array}{r}
 0x45673040 \\
 2C \\
 \hline
 306C
 \end{array}$$

$$\begin{array}{r}
 0x45673040 \\
 8 \\
 \hline
 3048
 \end{array}$$

5. Interrupt question #1: This question deals with setting up interrupts. In the space below provide code for the trainer board system to set up two I/O interrupts (UART and first timer, so second and fourth LSBs) and the PIT interrupt. The code should set up the system to be sensitive to the three interrupts. Also, initialization of the PIT interrupt should set a value to zero; this value will be used later when the ISR is invoked. PIT count value is 0x22222, timer count value is 0x33333. (PIT SPR is 987)

lis r1, 0x33333  
 Set IER  
 Set MER  
 lis r2, 0x81480000  
 SET CTR  
 lis r3, 0x81440000  
 lis r4, 0x81450000  
 Set IPR  
 Set ESR  
 Set CTIE  
 lis r8, 0x2222)  
mtspr r8  
 wrq, 0x10 link?  
stw r9, IAR(r1)  
 lis r9, 0x8 A  
stw r9, IER(r1)  
stw r9, IAR(r1)  
 lis r9, 0x0  
stw r9, 4(r3)  
stw r9, 4(r4)  
writeli #  
stw

not done



6. Interrupt question #2. This question deals with the steady state activities associated with interrupts set up in Question 5. On this page, give the code for Interrupt Service Routine activities for the interrupts in question. The UART should echo the character. The timer interrupt should increment (by one) the value stored at 0x33300. And the PIT activity should complement the value established in the initialization and send it out to address 0x81440000.

lis r1 0x3333  
trs r2 0x81440000  
. Set RXFIFO

lis r3 0x33300

here : b here

lwz r4, RXFIFO  
stw r4, (0&r3) IAR  
li r5 0x8430000  
stw r5, IAR(r3)

rfi

7. A user has collected data over an extended period. This data has been organized in a data structure that is an array of records. Each record is organized as shown at right: the value (in the first word slot), followed by the time (two 32 bit words from time base) and a sensor number (another 32 bit word). The user has collected 4096 of these records, and they are stored in consecutive memory locations starting at address 0x60000. In the space provided below, present a code fragment that will determine the average value of the entries.

First word	Value
Second word	Time (MS half)
Third word	Time (LS half)
Fourth word	Sensor Number

```

.org 0x10000
lis r1 0x60000
ori r1,r1,0x60000?
lis r2,4096
li r3,0
li r4,0
li r5,0
li r6,0
li r7,0
li r8,0
li r9,0
li r10,0
loop:
    add r4,r5,r6
    add r3,r7,r8
    add r10,r4,r3
    divid. r10,r10,4
    bdnz loop
where is address adjustment?

```

8. Let's try this again. Let R1 be the stack pointer. Present two routines, one for PUSH and one for POP. The Push routine should take the current return address, place it into R31, and then push onto the stack registers R16 thru R31. The POP removes from the stack the 16 register values and restores the values to appropriate registers, then restores the return address. Make sure that the stack pointer is dealt with appropriately.

PUSH

```

addi r1, r1, -68
stw r31, 60(r1)
stw r30, 56(r1)
stw r29, 52(r1)
stw r28, 48(r1)
stw r27, 44(r1)
stw r26, 40(r1)
stw r25, 36(r1)
stw r24, 32(r1)
stw r23, 28(r1)
stw r22, 24(r1)
stw r21, 20(r1)
stw r20, 16(r1)
stw r19, 12(r1)
stw r18, 8(r1)
stw r17, 4(r1)
stw r16, 0(r1)

```

mflr   
blr target

R16 → 31



R16 → 31

Need 68 places

Stmw!!POP

docmd r11  
lwz r11, 0(r1)  
mtfr r11

```

lwz r16, 0(r1)
lwz r17, 4(r1)
lwz r18, 8(r1)
lwz r19, 12(r1)
lwz r20, 16(r1)
lwz r21, 20(r1)
lwz r22, 24(r1)
lwz r23, 28(r1)

```

Ldmw!!

```

lwz r24, 32(r1)
lwz r25, 34(r1)
lwz r26, 36(r1)
lwz r27, 40(r1)
lwz r28, 44(r1)
lwz r29, 48(r1)
lwz r30, 52(r1)
lwz r31, 56(r1)

```

addi r1, r1, 68  
blr

Table 4: XPS UART Lite Registers

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x0	Rx FIFO	Read	0x0	Receive Data FIFO
C_BASEADDR + 0x4	Tx FIFO	Write	0x0	Transmit Data FIFO
C_BASEADDR + 0x8	STAT_REG	Read	0x4	UART Lite Status Register
C_BASEADDR + 0xC	CTRL_REG	Write	0x0	UART Lite Control Register

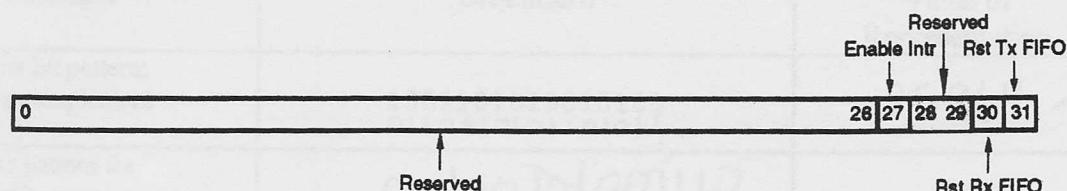


Figure 4: UART Lite Control Register

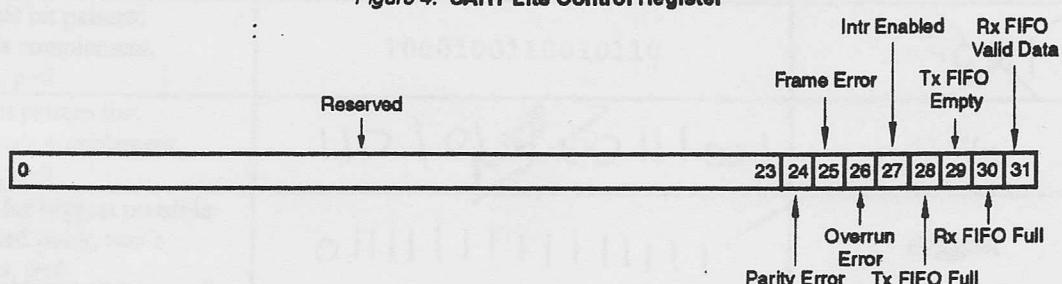


Figure 5: UART Lite Status Register

The addresses of the XPS Timer/ Counter registers are shown in the Table 4:

Table 4: XPS Timer/Counter Register Address Map

Register	Address (Hex)	Size	Type	Description
TCSR0	C_BASEADDR + 0x00	Word	ReadWrite	Control/Status Register 0
TLR0	C_BASEADDR + 0x04	Word	ReadWrite	Load Register 0
TCR0	C_BASEADDR + 0x08	Word	Read	Timer/Counter Register 0

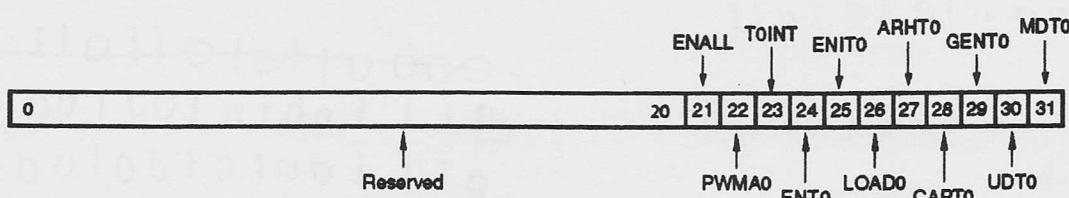


Figure 6: Timer Control/Status Register 0 (TCSR0)

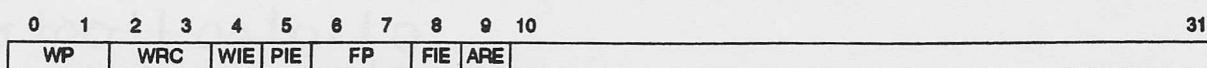


Figure 8-4: Timer-Control Register (TCR0)

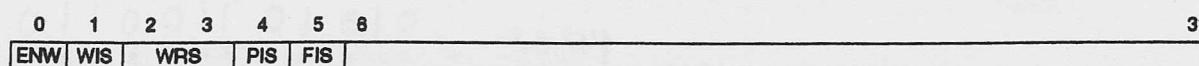


Figure 8-5: Timer-Status Register (TSR)

ECE-331 Final Exam Solutions

Two's Complement Pattern	1001100110010000	-103.34375
Unsigned Binary Pattern	1001100110010000	152.5625
Two's Complement Pattern	1011101110000000	24,000
Bit Two's Complement	1111100010101000	-7.34375
Bit Two's Complement	1101001001111000	-45 17/32
Unsigned Binary	1010101010101010	170.664063

1.

2.

a) fundamental principle on which stored program computers are based:

Fetch, decode, execute

b) how is Memory Mapped I/O implemented?

I/O activity is carried out by writing and reading locations in memory.

I/O elements are assigned addresses in the memory space of the processor

c) what instructions move multiple register values to and from memory?

lmw, stmw

d) vector table starts at 0x00100000. ISR is created to handle privilege violations,

where should the ISR be located?

0x00100700

e) give a two instruction sequence to look for the third legal location of interrupt table

lis r3, 0x0002

mtevpr r3

f) why does PowerPC have both a user mode and a privilege mode?

for protection and to manage the resources in the operating system

g) how does the system enter user mode?

Setting PR bit to 1, in the MSR

```
li r2 0x00010000 mtspr msr r2
```

h) how far can a branch-to-subroutine go? Assume instruction is at 0x00010000.

$$0x07fffffc + 0x00010000 = 0x000100ff$$

3. registers changed:

r1 = 0x11111118

r2 = 0x00000008

r7 = 0xFFFFFFFF

r9 = 0xBBBBBBBB

r10 = 0x12345678

LR = 0x3008

4. registers changed:

r0 = 0xDDDDDDDD

r3 = 0x54321065

r4 = 0xFFFFFFFF

r13 = 0xFFCCCCCC

r14 = 0x40000400

r15 = 0x0000EFFE

memory changed:

address: 0x4000042c → 99 99 99

address: 0x40000432 → 88

5. Interrupt Question #1: Provide code to configure the ML403 as we had in the lab, but fixed

so that it only acts if the interrupt is from the UART.

lis r1, 0x83e0  
lis r2, 0x80  
stw r2, 0x100C(r1) #LCR  
li r3, 0x45  
stw r3, 0x1000(r1) #DLL  
li r3, 0x01  
stw r3, 0x1004(r1) #DLM  
li r3, 0x7  
stw r3, 0x100C(r1)  
li r3, 0x8  
stw r3, 0x1004(r1)  
lis r4, 0x8140  
li r0, 0  
stw r0, 4(r4)  
lis r6, 0x8141  
stw r0, 4(r6)  
lis r7, 0x8142  
stw r0, 4(r7)  
lis r8, 0x8143  
stw r0, 0(r8)  
lis r10, 0x8180 #INTC  
li r1, 0x3  
stw r1, 0xC(r10) #MER  
li r1, 0x8  
stw r1, 0xC(r10) #IER  
li r9, 0xF

stw r9, 0(r10) #ISR

lis r22, 0020

mtevpr r22

wrteei 1

b here

6. Interrupt Question #2: deals with steady state activities from Question 5; give code for the ISR

for the IAR PIF

lis r6, 0x8180 #INTC

lis r5, 0x83c0 #UART

.org 0x500

lwz r1, 0x1000(r5) #load from RPR

lis r2, 0xffff

ori r2, r2, 0x0100

stw r1, 0(r2)

li r3, 0x60

stw r3, 0x1014 #LSR

li r3, 0x8

stw r3, 0xC(r6) #IAR

rfi

7. Provide two sets of instructions, one for the “push” operation and one for the “pop” operation

Push:

.org 0xhe11 (for Mike)

lis r1, 0x2000

stw r3, 0x10(r1)

stw r8, c(r1)

stw r14, 8(r1)

stw r22, 4(r1)

stw r25, 0(r1)

Pop:

.org 0x0200

lwz r25, 0(r1)

addi r1, r1, 4

lwz r22, 4(r1)

addi r1, r1, 4

lwz r14, 8(r1)

addi r1, r1, 4

lwz r8, c(r1)

addi r1, r1, 4

lwz r3, 0x10(r1)

8.

a) What is the instruction represented by the bit pattern 0x38857342?

addi r4, r5, 0x7342

b) what is the instruction represented by 0x4001511 located at 00004000?

bl 0x1510

0x1510 away from 0x00004000

c) give the coding for the instruction xor, r3, r4, r5

0x7C832A79