1. Information representation. We gotta have some question about number representation. Consider number system(s) that contain 10 bits, with the radix point to the right of the second bit (starting at the MSB). That is, there are two bits, then a radix point, then 8 more bits (xx.xxxxxxxx). For that arrangement of bits, fill in the missing elements of the following table. (Remember that Maximum is right-most on the number line; minimum is left-most on the number line.) Oh, and for the last line, provide the Value for the given bit pattern. Don't worry about turning the fraction version into a decimal version – just leave it as an integer-part, fraction-part answer.

Value	Unsigned binary pattern	Twos-complement pattern
Maximum	11.1111111111 ✓	01.1111111111 ✓
Minimum	00.0000000000 ✓	10.0000000000 ✓
$1\ ^{5}/_{32}$	01.00101000 ✓	01.00101000 ✓
$-1\ ^{9}/_{16}$	N/A	11.10010000 NO
1¾	01.11000000 ✓	N/A
$-1\ ^{11}/_{32}$	N/A	1010101000

Correct answer: 10.01110000

2. General information question:
   a) Give a two instruction sequence to set register R16 to the value 0x12345678.

```
lis R16, 0x1234
ori R16, R16, 0x5678
```

   b) Give a sequence of instructions (only 2 needed) that will set up the system to expect the interrupt table to be found at the third legal location for the table. That is, what is the third legal location for the interrupt table, and how do you set it up?

```
lis r1, 0x0002
mtspr evpr, r1
```

Must be multiple of 64k.

   c) When a branch-to-subroutine is encountered, where does the system store the address to which the subroutine should return?

link register

   d) Assume a conditional branch is located at address 0x00100000. What is the highest address that can serve as the target of the branch? That is, what is the highest address that can be reached?

```
  0x00100000
+ 0x000FFFFF
  0x001FFFFF   NO (-4)
```

Correct Answer: 0x00107FFC

   e) The FIT can provide one interval out of how many choices?

4

3. Interrupt Controller Question: In the table below, identify the registers that need to be initialized, and give values for each. In this interrupt system, there are four interrupt sources, starting in the least significant bit position, and all are to be enabled. Also, the software activation of interrupts is not to be utilized. Assume that you want to assert the appropriate bits to reset any flags that may have remained from an earlier program execution.

Addr Offset	Register	Bit Pattern
0x00	ISR	0x 0000000F not writable
0x04	IPR	read - only
0x08	IER	0x 0000 000F
0x0C	IAR	0x 0000 000F
0x1C	MER	0x 3

N/A

Now, in the space provided below, give instructions that will establish the bit patterns given above as well as to a)set up the EVPR register (to `0x000A0000`) and b) set up any enabling activity needed to allow interrupts in general. Assume that the interrupt controller has been located at address `0x80440000`.

```
.set ISR, 0x00
.set IPR, 0x04
.set IER, 0x08
.set IAR, 0x0C
.set MER, 0x1C

li r0, 0
lis r5, 0x8044          # load address for ICONT
Lis r6, 0x000A          # load evpr addr.
li r7, 0xF
li r8, 0x3              # bit pattern for MER
stw r0, 0(r5)           no!
stw r7, IER(r5)
stw r7, IAR(r5)
stw r8, MER(r5)
mtspr evpr, r6
wrteei 1
```

4. ISR question: For system that utilizes the PIT to create a periodic time function, give code for an Interrupt Service Routine for the following situation. When the PIT module causes an interrupt, reset any appropriate flags, increment the value in R27 and send the value to the LEDs. The LED interface GPIO is located at **0x84480000**. Do not worry about register volatility.

```
.set 0x500 why?          # start for non-crit ints
lis r7, 0x8448

li r0, 0
li r1, 0x0F              # Non-zero value for m-box
stw r0, 4(r7)            # set up LEDs as output
li r9, 0x8               # stroke bit 5 in TSR  wrong bit
mttsr r9
lis r8, 0x6000           # pointer to mailbox

stw r0, 0(r8)

check: lwz r6, 0(r8)

cmpwi 0, r6, 0
bf 2, reset ?

reset: stw r0, 0(r8)     # reset m-box

li r27, 0
addi r27, r27, 1

stw r27, 4(r7)?          # send value to LEDs
```

Correct Answer

where is rfi?

pitcode:

```
lis r10, 0x0800
mttsr r10
lis r24, 0x8448
addi r27, r27, 1
stw r27, 0(r24)
rfi
```

5. Data structure question. Consider the situation where a user has an array of halfwords. The operation that is to be performed on this array of data is to sum all of the elements. The array starts at address **0xFF000100** (and progresses to higher addresses). Create code that will calculate the sum of the first 800 values of this array. Note: you might make use of the sign-extension instructions shown in material at end of test.

sthw

```
.org 0x3000

lis r1, 0xFF00          # set up pointer to array
ori r1, r1, 0x0100

li r0, 0
li r2, 800              # load ctr value
mtctr r2                # putting 800 in ctr

sum: lhz r3, 0(r1)      # load first element
     add r0, r0, r3     sign extend!
     addi r1, r1, 2     # incr to next halfword in array

     bdnz sum
```

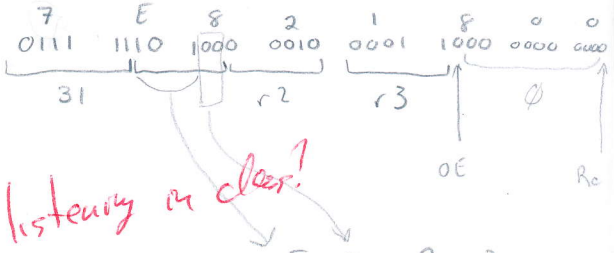
extsh r3, r3
sign extend value

good job

6. Instruction coding question:

a) What is the instruction represented by the bit pattern **0x7E821800**?

Correct Answer:



cmp 5, 0, r2, r3

Don't know how to go from op code to instruction and vice versa.

so - weren't you listening in class?

(-5)

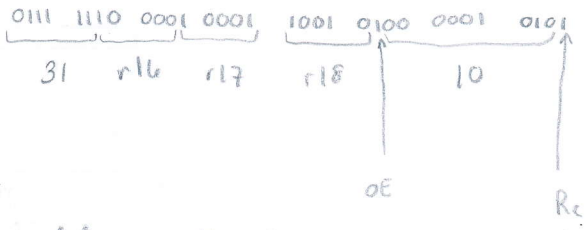
b) What is the instruction represented by the bit pattern **0x7E119415**?

addis r4, r9, 11

addco. r16, r17, r8

(-4)

Correct Answer:



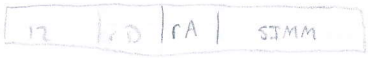
addco. r16, r17, r18

c) Give the coding for the instruction **addic r2,r3,r4**

Correct Answer:

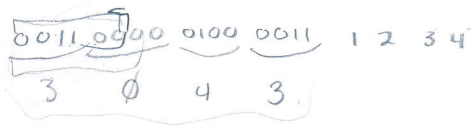
rD => r2

rA => r3



0x

(-5)



0x 30431234

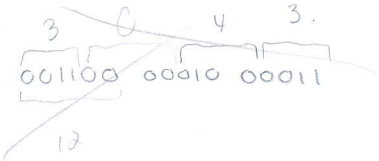


Table 4: **XPS INTC Registers and Base Address Offsets**

Register Name	Base Address + Offset (Hex)	Access Type	Abbreviation	Reset Value
Interrupt Status Register	C_BASEADDR + 0x0	Read / Write	ISR	All Zeros
Interrupt Pending Register	C_BASEADDR + 0x4	Read only	IPR	All Zeros
Interrupt Enable Register	C_BASEADDR + 0x8	Read / Write	IER	All Zeros
Interrupt Acknowledge Register	C_BASEADDR + 0xC	Write only	IAR	All Zeros
Set Interrupt Enable Bits	C_BASEADDR + 0x10	Write only	SIE	All Zeros
Clear Interrupt Enable Bits	C_BASEADDR + 0x14	Write only	CIE	All Zeros
Interrupt Vector Register	C_BASEADDR + 0x18	Read only	IVR	All Ones
Master Enable Register	C_BASEADDR + 0x1C	Read / Write	MER	All Zeros

Timer-Control Register

The timer-control register (TCR) is a 32-bit register used to control the PPC405 timer events. Figure 8-4 shows the format of the TCR. The fields in TCR are defined as shown in Table 8-3.

Figure 8-4: **Timer-Control Register (TCR)**

Timer-Status Register

The timer-status register (TSR) is a 32-bit register used to report status for the PPC405 timer events. Figure 8-5 shows the format of the TSR. The fields in TSR are defined as shown in Table 8-4.

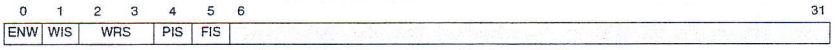


Figure 8-5: **Timer-Status Register (TSR)**

Table 3-32: **Sign-Extension Instructions**

Mnemonic	Name	Operation	Operand Syntax
Extend-Sign Byte Instructions		rA[24:31] is loaded with (rS[24:31]). The remaining bits rA[0:23] are each loaded with a copy of (rS[24]).	
extsb	Extend Sign Byte	CR0 is *not* updated.	rA,rS
extsb.	Extend Sign Byte and Record	CR0 is updated to reflect the result.	
Extend-Sign Halfword Instructions		rA[16:31] is loaded with (rS[16:31]). The remaining bits rA[0:15] are each loaded with a copy of (rS[16]).	
extsh	Extend Sign Halfword	CR0 is *not* updated.	rA,rS
extsh.	Extend Sign Halfword and Record	CR0 is updated to reflect the result.	

Table B-1: **Instructions Sorted by Mnemonic**

	0	6	9	11	12	14	16	17	20	21	22	26	30	31
add	31	rD		rA			rB			OE	266			Rc
addc	31	rD		rA			rB			OE	10			Rc
adde	31	rD		rA			rB			OE	138			Rc
addi	14	rD		rA			SIMM							
addic	12	rD		rA			SIMM							
addic.	13	rD		rA			SIMM							
addis	15	rD		rA			SIMM							
addme	31	rD		rA			00000			OE	234			Rc
addze	31	rD		rA			00000			OE	202			Rc
and	31	rS		rA			rB				28			Rc
andc	31	rS		rA			rB				60			Rc
andi.	28	rS		rA			UIMM							
andis.	29	rS		rA			UIMM							
b	18	LI											AA	LK
bc	16	BO		BI			BD						AA	LK
bcctr	19	BO		BI			00000				528			LK
bclr	19	BO		BI			00000				16			LK
cmp	31	crfD	00	rA			rB				0			0
cmpi	11	crfD	00	rA			SIMM							
cmpl	31	crfD	00	rA			rB				32			0