EECS 373 Exam 1

Winter '04, Prof. Mark Brehob

Name: Key uname: Key

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Section	Points
Short answer	/60
Design question	/40
Total	**/100**

NOTES:

- Open notes. If you don't have your white book with you please contact the proctor – you will need it!
- ***Don't spend too much time on any one problem.***
- You have 80 minutes for the exam.

1. If SIPEND=0x2C010000 and SIMASK=0x5F000000 what is the value in SIVEC? Provide your answer as a 16-bit hex number. **[5]**

SIPEND = 0010 1100 0000 0001

SI MASK = 0101 1111 0000 0000

AND = 0000 1100

IRQ0 LVL 0

IRQ2 LVL 2

SIVEC from table = 00010000 2

0x1000

2. Say we were using code as found in the *rightmost* part of figure 12-3 on page 12-11 of the white book. If BASE=0x00340000, where would the code for the LVL3 interrupt be located? **[5]**

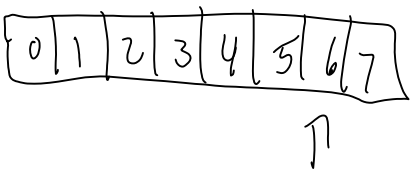
LVL3 = 0001 1100 from table. = 1C

Loading as 1/2 word = 1C00. So address = 0x00341C00

3. Write PowerPC assembly that modifies the byte stored at memory location 0x00001000 so that bit 6 is negated (changed from a zero to a one or visa versa). No other bit in memory should be changed by your code. **[10]**

```
lba  R1, 0x1000(R0)
xori R1, R1, 2
stb  R1 0x1000 (R0)
```

Note: in PPC (big endian) bit 6:



4. When an interrupt occurs, SRR0 and SRR1 are updated automatically by the hardware. The general philosophy of the MPC823 with respect to interrupts is to do as little as possible automatically, leaving it to the ISR to handle the rest (such as saving registers to the stack.) Why couldn't the values stored into SSR0 and SRR1 instead be saved by the ISR? **[5]**

SRR0 holds the old PC value. (Place to return to).

SRR1 holds old value of MSR.

Both PC + MSR change on an interupt. So if the hardware didn't save them you would lose the (important) information.

5. Write the prologue for a PowerPC assembly function "bob". You are to follow the EABI and use as little memory on the stack as possible. The function "bob" uses registers 4, 5, 30, and 31 but does not modify the condition register. Other than saving the needed GPRs, 8 bytes need to be reserved for local variables. **[10]**

Need to save Return address, Back pointer, 2 registers (30 & 31) as well as 8 bytes for local. Thus 24 bytes.

```
stwu   R1, -24(R1)
mflr   R2
stw    R2, 28(R1)
stwm   R30, 16(R1).
```

6. For each of the load instructions below, indicate whether or not the instruction causes an unaligned access. Also, list the bus contents for each transaction and the final value held in the register that you've loaded. If there are bits that have undefined values for a given transaction, mark their values as 'X'. Use hexadecimal notation. Assume the register and memory values listed below. **[15]**

Register	Value
r3	0x03100002

Memory Location	Value
0x03100000	0x12345670
0x03100004	0x14151617
0x03100008	0xDEADBEEF
0x0310000C	0x0BADBEAD
0x03100010	0x00FADDAD
0x03100014	0x44556677
0x03100018	0x37337337
0x0310001C	0x27037000

Bus contents could be of more than 6 he form....

Assembly command	Aligned?	Bus contents	Value in r8 once assembly command is done
lha r8, 7(r3)	N	XXAD XXXX XXXX BEXX 	0xFFFFADBE
lwz r8, 3(r3)		XX15 XXXX XXXX1617 DEXXXX	0x15161 7DE

7. The memory locations 0x1000 through 0x1006 have been initialized as shown in the table below and the registers have been initialized so that r1=1, r2=0x1000, and r3=0x01234567. The following code segment is then run.

```
stw  r3, 0(r2)
stbu r3, 2(r2)
stwx r3, r2, r1
```

notice update!

After the code segment completes, what are the values in these memory locations? You may ignore any memory location that is written to but not shown in the table. **[10]**

Address	**0x1000**	**0x1001**	**0x1002**	**0x1003**	**0x1004**	**0x1005**	**0x1006**
Previous value (hex)	A0	B0	C0	D0	E0	F0	A0
New value (hex)	01	23	67	01	23	45	67

1st 01 23 45 67

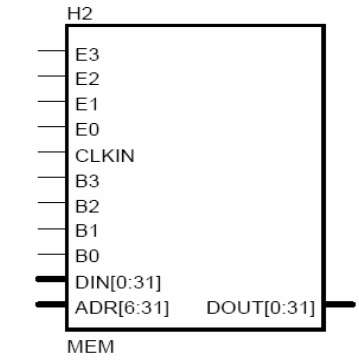
2nd 67

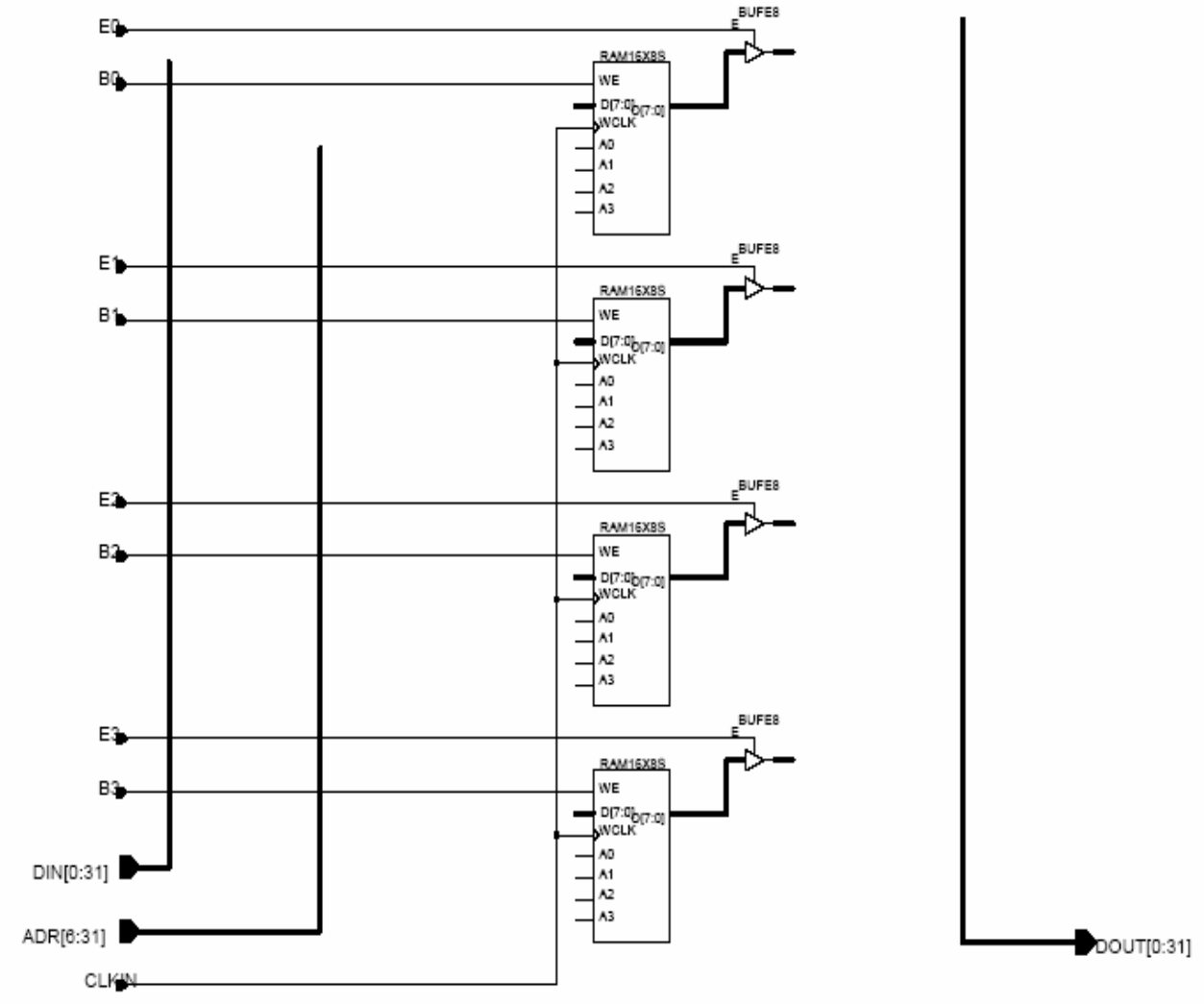
3rd 01 23 45 67

Design Question

Complete the design of the following MPC823 memory. The memory accommodates byte, ½ word and word read and write accesses. The memory has a 32-word capacity and is composed of 16 X 8-bit RAM modules. Assume that the memory is in the address space 0x0000 007f to 0x0000 0000.

1) Complete the connections for the following macro. Be sure to label all your bus connections clearly. **[10]**





2). You are given two macros that decode A31-A30 and TSIZ[0:1] according to the tables found below (First column is inputs, rest are outputs)

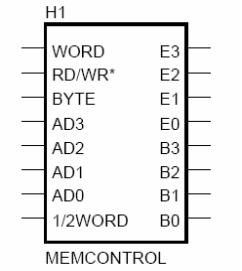
ADS_MACRO

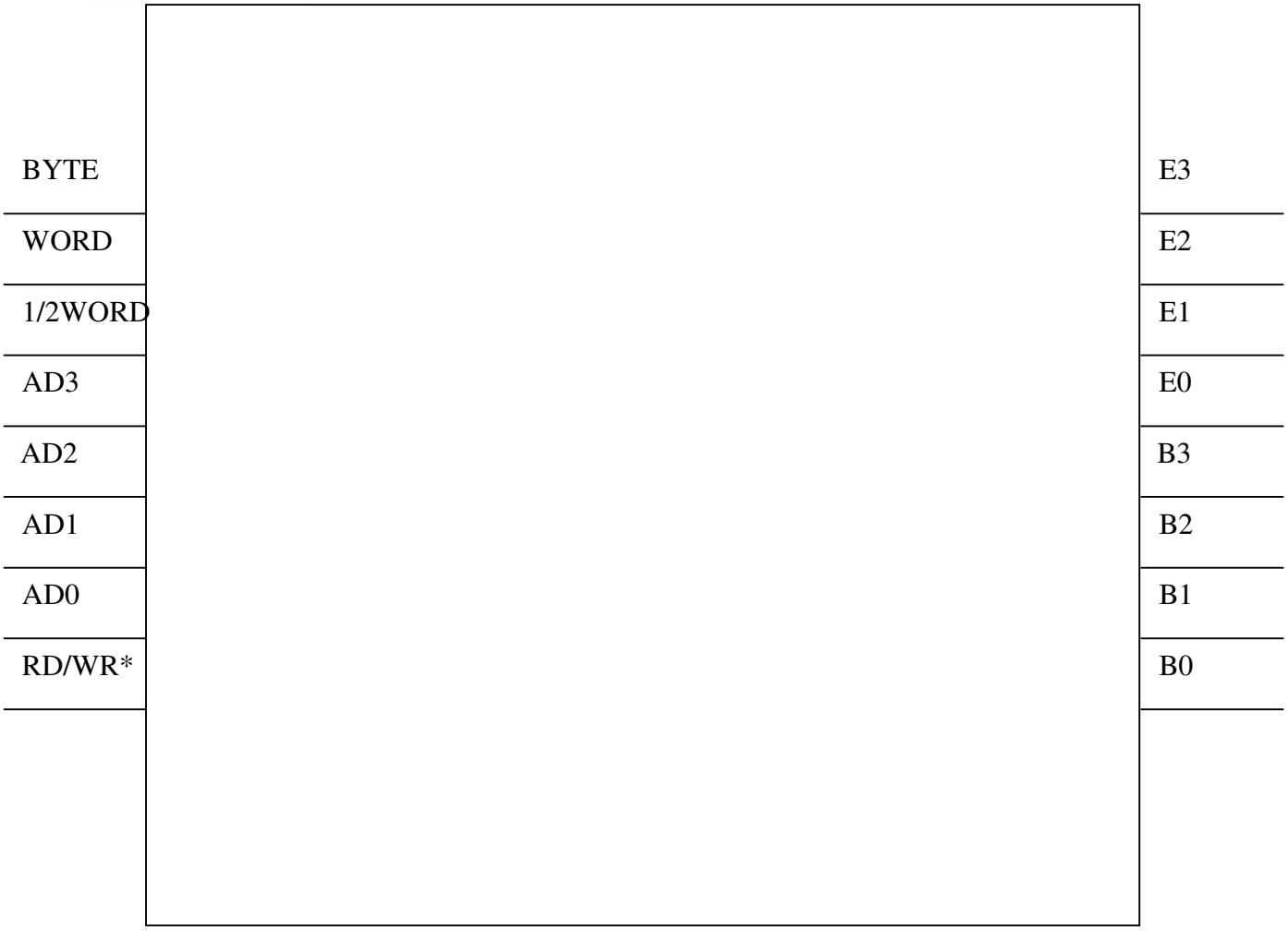
A30, A31	AD0	AD1	AD2	AD3
0,0	1	0	0	0
0,1	0	1	0	0
1,0	0	0	1	0
1,1	0	0	0	1

TSIZE_MACRO

TSIZ[0:1]	byte	1/2word	word
0,0	0	0	1
0,1	1	0	0
1,0	0	1	0

Design a macro to provide the control logic for byte, ½ word and word read and write access for the proceeding module. The macro should have the following hierarchical connections. Use the space provided below. **[20]**





3. Now complete the design by connecting the macros. You may need to add logic. You do **NOT** need to worry about: **[10]**

- The generation of TA
- The decoding of the address
- The inputs into the H1 macro. (Just leave them unconnected)
- The CLKIN wire in the H2 macro (Just leave it unconnected)