EECS 373 – Winter 2001
Midterm 1

READ AND FOLLOW THESE INSTRUCTIONS:

1. Do not begin until you are told to do so.
2. You have 75 minutes; budget your time. The questions are not of equal weight; do not spend too much time on a question that is not worth many points.
3. Read through the entire question before beginning it.
4. This exam is **closed notes**. You may use the MPC823 data book, the "PowerPC Programming Pocket Book", and/or a printout of Appendix F from the green book as reference material. You may *not* share reference materials with other students.

Name: Solution

Student ID: _____
(not your SSN)

Sign the Honor Code Pledge:

I have neither given nor received aid on this exam, nor have I concealed any violation of the Honor Code.

Signature: _____

Section	Points	Score
1	18	
2	17	
3	30	
4	35	
Total	100	

Section 1: Short Answer (3pts each)

1. What purpose(s) does an ABI serve?

   1) Standardized procedure interfaces

2. Explain the difference between a little-endian machine and a big-endian machine. Which is the Power PC?

   A little-endian machine stores the LS byte of a word first (lowest byte address), vs. a big-endian machine which stores the MS byte first.

3. How do you define an unaligned memory access? Why do we prefer to do aligned data transfers as opposed to unaligned transfers?

   Any access where the starting address is not a multiple of 2^(transfer size-1)

4. Define "Setup Time" and "Hold Time".

   Setup Time → The time a signal must be present before a clock edge in order to be latched properly.
   Hold Time → The time a signal must remain present after a clock edge in order to be latched properly.

5. What is the purpose of the "Prologue" code we place at the beginning of a function? When do we need it? When do we not need it?
   To create a stack frame.
   So we have a place to store the LR (function args & temp variables)
   We don't strictly need it for leaf functions.

6. Describe what (if any) transaction will be noticed on the microprocessor bus for the following instructions:

   a. `extsh  r2, r0`
      → NONE

   b. `lis  r1, stack`
      → NONE

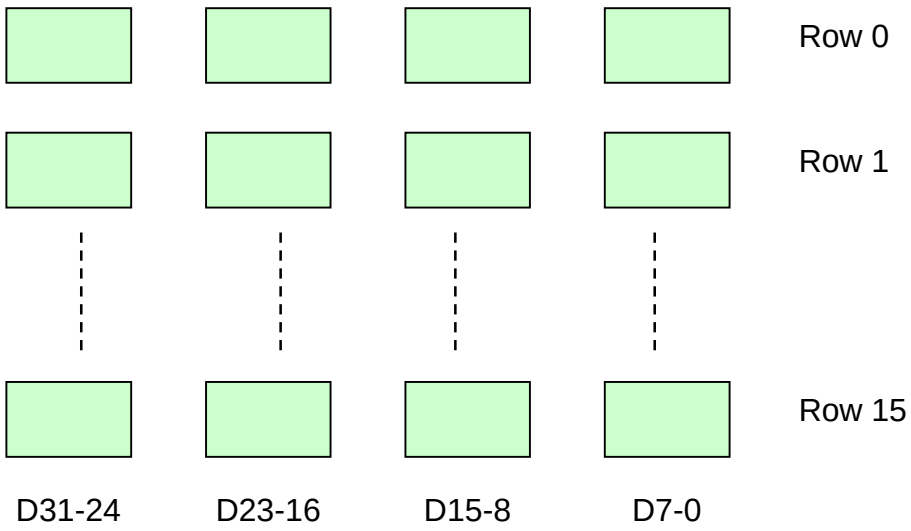
   c. `lwarx  r7, r7, r3`
      → Loads a word from memory address (r7 + r3)

Section 2: Memory Arrays

7. Given an unlimited supply of 16Kbit memory chips (organized as 2048x8):
   a. (3 pts) How many are needed to create a 32K-word memory array if a "word" is 4 bytes?

      (2Kx8) * 4 → 2Kx32; (2Kx32) * 16 → 32Kx32…. 64 chips

   b. (14 pts) Sketch this memory array. Assume that the LS-bit (LS ≡ Least Significant) of the address and data buses is labeled bit 0. Indicate which address lines will be required at each memory chip. Indicate which data bus lines are attached to each memory chip. Indicate (in general terms) the control signals and address lines needed by the decoder for this memory bank.



A1-A0	Use in decoding circuitry to choose column
A12-A2	Go to all chips
A16-A13	Use in decoding circuitry to choose row

Section 3: PowerPC Assembly Language

8. (10 pts) Give an instruction or sequence of instructions that reads the word at memory location 0x346000. If bit 27 of the word is zero, branch to label `loop`. If it's not, read the word at memory location 0x346008 and branch to label `loop` if the 27th bit of that word is set.

```
lis     r2, 0x34
ori     r2, r2, 0x6000
lwz     r5, 0(r2)
rlwinm. r4, r5, 0, 27, 27
beq     loop
lwz     r6, 8(r2)
rlwinm. r4, r6, 0, 27, 27
bne     loop
```

9. Each of the following PowerPC assembly code fragments is supposed to perform some function, as described by the comments. However, each fragment contains at least one error and therefore does not function as expected. Explain the problem(s) you find, and the changes necessary to make the code function as described.

(a) (5 pts)

```
        and  r5, r5, r0          # Clear r5
loop:   addi r5, r5, 1           # Increment r5
        cmpdi r5, 100            # Compare r5 to 100
        blt  loop                # If r5 < 100, branch
                                 # back to loop
```

- R0 is not necessarily zero for the AND instruction.

(b) (5 pts)

```
        li   r4, r4, 8           # Load r4 with value 8
loop:   subf r3, r3, r4          # Decrement r3 by 8
        cmpdi r3, 0              # If r3 != 0
        b    loop                # branch back to loop
```

- the `subf` instruction should be either `subf r3,r4,r3` or `sub r3,r3,r4`
- the b instruction should be `bne`

(Question 9, continued)

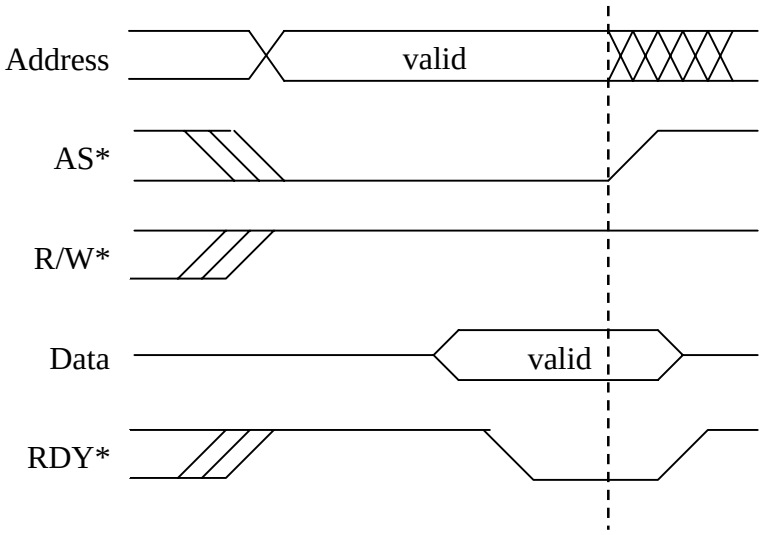
(c) (10 pts)

```
        .data
        .align 2
array:  .half 35, 14, 97, 12, 15, 100

             .text
             .align 2
             .global _start

_start: li   r2, array@h          # Load array address
        ori  r2, r2, array@l      # into r2
        li   r3, 15               # Load r3 with value 15
rdwd:   lhz  r4, 0(r2)            # Read next half-word
                                  # from array
        cmpd r4, r3               # Compare r4 and r3
        bc   12, 10, done         # If r4==r3, branch to
                                  # to done
        addi r2, r2, 2            # Increment r2 to point
                                  # to next half-word
        b    rdwd                 # Branch back to rdwd
done:   stw  r2, 0(r6)            # Store the value of r2
                                  # to the address pointed
                                  # to by r6
```

- The first `li` should be an `lis`
- The bc uses bit 10, which is the EQ bit of CR2, but the cmpd uses CR0 by default, so the branch should read:
  `bc 12, 2, done`
- Each half-word of data is actually stored as a word in memory with the top 16 bits set to 0, so the `addi` should increment r2 by 4, not 2.

Section 4: Bus Interfaces (35 pts)

10. The timing diagram below depicts the bus read cycle for the "Goober" microprocessor. Note that there is no clock associated with this bus specification.



Definitions:

Address	Address bus (16 bits, MS bit is bit 15)
AS*	Address strobe. Indicates the start of a new bus cycle, and that the address bus holds valid data. [Output from microprocessor]
R/W*	(If you have to ask…)
Data	Data bus (8 bits, MS bit is bit 7) (MS ≡ Most Significant)
RDY*	Indicates that the I/O or Memory device has completed the bus transaction. [Input to microprocessor]

Specifications:

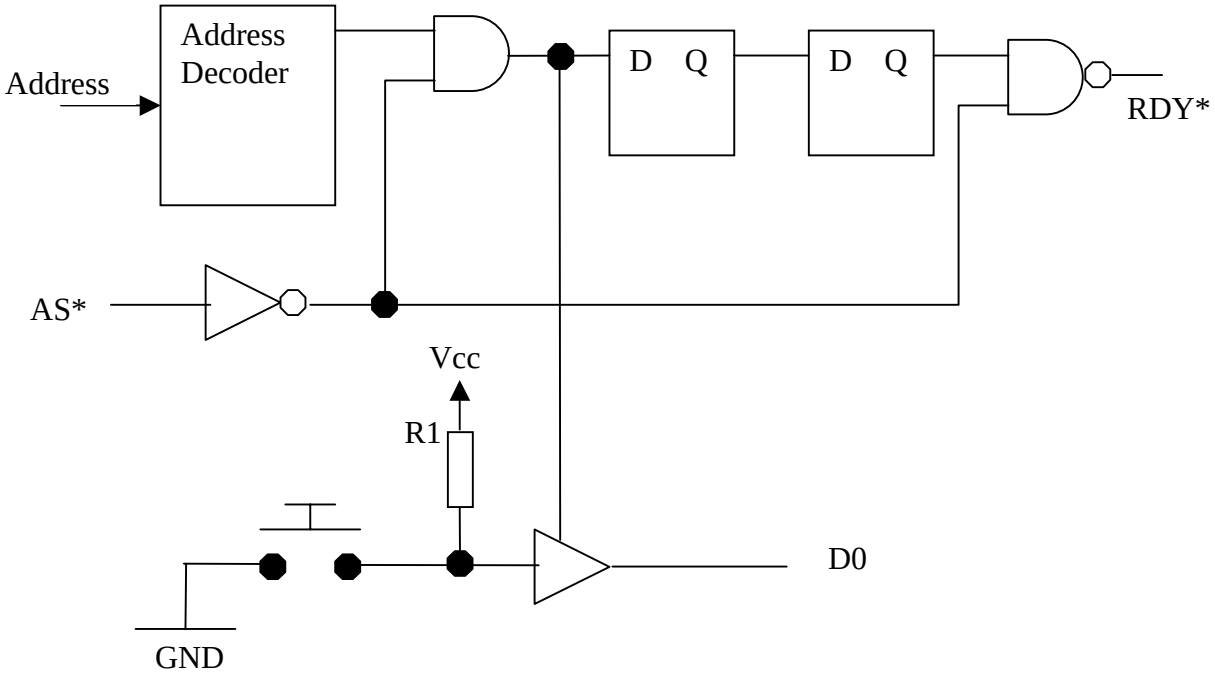
- The data on the data bus be valid for not less than 20ns prior to the assertion of the RDY* signal.
- The data bus and RDY* must be released (drivers turned off) not more than 20ns after AS* is removed by the microprocessor.

System Memory Map:

0xFFFF 0xF000	System ROM
0xEFFF 0x8000	{unused}
0x7FFF 0x0000	System RAM

(Problem 10, continued)

Design the circuitry necessary to read the state of a single switch input using the least-significant bit of the data bus. Assume that you have a 40MHz clock available, but that this clock is not synchronized to the Goober's bus. Referring to the system memory map above, use the minimum reasonable amount of circuitry to map the switch to location 0xC040. Specify any alias addresses for the switch.



- Need to decode A15 (A15=1)
- Need to decode two other bits in A14-12 (A14=1, A13=0, A12=0)
- Switch and pullup are not required for correct solution
- AS* used to remove data and RDY*, since clock is too slow (25ns) to remove RDY* within spec.
- Number of aliases:
  - N = number of address bits decoded
  - Number of matching addresses = 2^(16-N)