1. (20 points) Given the following data segment declaration, what values are printed by each of the syscalls?

```assembly
m:       .word    0xfc00ffcc
h:       .word    lc,rc
lc:      .word    13
rc:      .word    lgc,rgc
lgc:     .word    14
rgc:     .word    15
authors: .asciiz  "Patterson and Hennessey"
```

(a) `lb $a0,m`
(b) `lw $t6,m`
(c) `la $t0,authors`
(d) `la $t0,h`

**ANSWER:**

(a) `-4`
(b) `12`
(c) `"Patterson and Hennessey"`
(d) `Hennessey`

2. (30 points) Short answer.

(a) List the four design principles given in your text and discussed in class.

**ANSWER:**

1. Simplicity favors regularity.
2. Smaller is faster.
3. Good design demands good compromises.
4. Make the common case fast.
5. 

(b) What is the machine language instruction (given in hex) that corresponds to the assembly language instruction `lw  $t3,10($t2)`?

**ANSWER:** I format

```
<table>
<thead>
<tr>
<th>35</th>
<th>10</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>100011</td>
<td>01010</td>
<td>01011</td>
<td>00000000000001010</td>
</tr>
<tr>
<td>1000</td>
<td>1101</td>
<td>0100</td>
<td>1011</td>
</tr>
</tbody>
</table>
```

**ANSWER:** `-4         12`

```
<table>
<thead>
<tr>
<th>8</th>
<th>d</th>
<th>4</th>
<th>b</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---

*--- in hexadecimal
*--- in binary
*--- in binary, regrouped
*--- in hexadecimal
(c) Assume that $A$ is defined as an $n \times m$ integer matrix (meaning that it has $n$ rows and $m$ columns). Give an expression for the offset of element $(i, j)$ from the start of $A$, assuming that $A$ is stored in row major order and indexed starting at 0.

**ANSWER:** $(i \times m) + j \times 4$

(d) Assume that foo is a **procedure** that uses registers $s6$, $s7$, $t0$, and $t2$ and that foo calls one procedure, $g$, but that no register values are used across the call to $g$. Show the code needed to manipulate the stack on entry to foo if foo follows all standard conventions.

**ANSWER:**

```
sub     $sp,$sp,12
sw      $ra,0(4sp)
sw      $s6,4($sp)
sw      $s7,8($sp)
```

3. (25 points) Write a program segment that prints out the even-indexed elements of an array $A$ of 100 integers. Thus your segment should print the integer values of the elements $A[0]$, $A[2]$, $A[4]$, ..., $A[98]$. Show the declaration of $A$ in your .data section. You do not have to write a complete program, just the relevant part of the .data segment and the loop that prints the array elements. Print each element on a separate line.

**COMMENT YOUR CODE APPROPRIATELY.**

**ANSWER:**

```
.data
A:  .space 400        # space for array
Newline: .asciiz "\n"
...
li      $t0,0        # offset of first element
li      $t1,1400    # offset immediately after the last element
Loop:   beq      $t0,$t1,Go    # done printing
li      $v0,1      # no, print next element
lw      $a0,A($t0)
syscall
li      $v0,4      # print the end of line character
la      $a0,Newline
syscall
add     $t0,$t0,8  # increment for next element
j       Loop
Go:
```
4. (25 points) Write a recursive procedure, ReverseDigits, that prints the digits of a positive integer reversed. Each call should remove and print the rightmost digit of the integer and then recursively call ReverseDigits on what remains of the integer after its rightmost digit has been removed. If the procedure receives an argument of 0, it should just return.

Write a program that reads in a positive integer (you can assume its greater than 0) and then prints that integer with its digits reversed using your procedure. Make sure that ReverseDigits obeys all standard conventions.

Example: to print 598 in reverse, your main program should call ReverseDigits with 598
• the first invocation of ReverseDigits should print 8 and call ReverseDigits with 59
• the second invocation of ReverseDigits should print 9 and call ReverseDigits with 5
• the third invocation of ReverseDigits should print 5 and call ReverseDigits with 0
• the fourth invocation of ReverseDigits should just return

COMMENT YOUR CODE APPROPRIATELY. The answer to this problem is a complete program. You do not have to worry about prompting for input or labeling the output. Your program should terminate with an appropriate syscall.

ANSWER:

.data
.globl __start
.text
__start:   li $v0,5 # read integer to test procedure with
syscall
move $a0,$v0 # set up argument for ReverseDigits
jal ReverseDigits # call procedure
li $v0,10 # terminate
syscall
ReverseDigits:   sub $sp,$sp,4 # save return address on the stack
sw $ra,0($sp)
move $t0,$a0 # save parameter
beq $t0,$0,ReturnRD # see if done
div $t0,$t0,10 # no, remove rightmost digit
mfhi $a0 # print rightmost digit
li $v0,1
syscall
move $a0,$t0 # set up argument for recursive call
jal ReverseDigits
ReturnRD:   lw $ra,0($sp) # restore $ra for return
add $sp,$sp,4
jr $ra