Arquitectura de Computadores
543.426

Tarea No. 2
11 de abril de 2017

Resuelva los siguientes problemas utilizando la convención estándar de uso de registros MIPS (temporales, argumentos, valores de retorno, etc.). La eficiencia del código (velocidad, tamaño, uso de registros) también se considerará en la evaluación de la tarea.

Entregar un informe escrito (en computador) con la descripción de sus soluciones a cada uno de los problemas en secretaría de electrónica. Además deberá enviar al correo antonio.saavedra@openmailbox.org con las soluciones a cada problema. Los códigos serán probados utilizando el simulador MARS MIPS, por lo que no se corregirán programas que no puedan ser ensamblados en el mismo. Plazo máximo de entrega: Jueves 20 de Abril hasta las 17 horas. No se corregirán tareas atrasadas.

Se debe trabajar en grupos de 2 personas. Grupos de 1 persona son permitidos pero no recomendados. No se permitirán grupos de más de 2 personas. Se les recuerda que cualquier copia (entre tareas o de fuentes externas) resultarán en calificación 1 para todas las tareas involucradas.

Problema 1

La siguiente función en C calcula de manera recursiva el máximo valor entre los elementos de un vector v. Recibe como argumento un puntero al inicio de dicho vector y un entero indicando su largo, n.

```
1  int max(int *v, int n)
2  {
3    int max1, max2;
4
5    if (n < 2)
6      return v[0];
7
8    max1 = max(v, n/2);
9    max2 = max(v + n/2, n/2);
10
11   if (max1 > max2)
12     return max1;
13
14   return max2;
15 }
```

Asumiento que n es siempre un valor potencia de 2, escriba esta función en assembly MIPS. A continuación se entrega un código en MIPS que implementa un código main que verifica esta función con un vector de prueba definido en la sección de datos. Utilice este programa para verificar el funcionamiento de su código.

```
1  .data
2  N:       .word 8
3  vect:    .word 1, 7, 3, 2, 4, 6, 10, 8
4  printf:  .asciiz "El valor maximo es "
5  fin:     .asciiz "\n"
6
```

```
.text

######## FUNCION MAIN ########

main:   la   $a0, vect    # vec como argumento a MAX
        la   $a1, N       # direccion de N
        lw   $a1, 0($a1)  # N como argumento a MAX
        sw   $fp, -4($sp) # guarda el valor de fp en stack
        addi $fp, $sp, -4 # fp como la nueva base del stack
        sw   $ra, -4($fp) # guarda ra en stack
        addi $sp, $fp, -4 # sp como el tope del stack
        jal  MAX          # MAX(vect,N)
        addi $sp, $fp, 4  # recupera el tope del stack
        lw   $ra, -4($fp) # recupera ra
        lw   $fp, 0($fp)  # recupera fp
        move $t0, $v0     # el valor maximo es guardado
        li   $v0, 4       # se configura el syscall para imprimir string
        la   $a0, print   # se carga el string a imprimir
        syscall           # se imprime string
        li   $v0, 1       # se configura el syscall para impimir entero
        move $a0, $t0     # N para imprimir
        syscall           # se imprime N
        li   $v0,4        # se configura el syscall para imprimir string
        la   $a0, fin     # se carga el string a imprimir
        syscall           # se imprime string
        li   $v0, 10      # se configura el syscall para finalizacion
        syscall           # termina programa

######## FUNCION MAX ########

MAX: ## INGRESE SU CODIGO ###
```

Problema 2

En los campos de la lógica y las matemáticas se conoce una secuencia de Hofstadter a una familia de secuencias de enteros definidos a partir de relaciones de recurrencia no lineales. En particular la secuencia de Hofstadter masculina y femenina se define según la siguiente relación:

$$\begin{aligned} F(0) = 1; \quad M(0) = 0 \\ F(n) = n - M(F(n-1)), \quad n > 0 \\ M(n) = n - F(M(n-1)), \quad n > 0 \end{aligned}$$

Las siguientes funciones en C permiten calcular el n-ésimo número de estas secuencias a partir de llamdas mutuamente recursivas.

```
1  int F(const int n);
2  int M(const int n);
3
4  int F(const int n)
5  {
6    if (n == 0)
7      return 1;
8    else
9      return n - M(F(n - 1));
10 }
11
12 int M(const int n)
13 {
14   if (n == 0)
15     return 0;
16   else
17     return n - F(M(n - 1));
18 }
```

Escriba un programa en assembly MIPS que sea capaz de implementar estas funciones siguiente el esquema de recursión mutua. Su programa deberá calcular los n primeros femeninos y masculinos de esta secuencia e imprimirlos en pantalla utilizando el syscall correspondiente. El valor de n deberá ser definido en la sección de datos del programa.

Problema 3

La siguiente linea de código es el protipo de una función en C:

```
1  int mat_sumspc (int *m1, int *m2, int *m3, int n);
```

Esta función debe realizar las siguientes operaciones sobre dos matrices de dimensiones $n \times n$: En primer lugar debe sumar ambas matrices. Luego debe verificar si el resultado de la suma es una matriz simétrica, para finalmente, en caso de no ser simétrica, calcular su transpuesta. La función debe retornar un 0 si el resultado era una matriz simétrica y un 1 en caso contrario. Adicionalmente la función guarda en el resultado de la transpuesta de la suma en la tercera matriz que recibe como argumento (no es necesario calcular explicitamente la transpuesta en caso de ser simétrica). El cuarto argumento contiene la información del largo de las matrices, n.

Implemente esta función a partir de lo descrito en lenguaje assembly MIPS.

Recuerde que en C las matrices se almacenan por filas en memoria y el nombre de la matriz es un puntero al primer elemento de ésta. En consecuencia, para una matriz de dimensiones N filas x M columnas, donde el puntero int *p apunta al primer elemento, la notación $p[i][j]$ es equivalente a $*(p + i * M + j)$.