ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

Organización y Arquitectura de Computadores

Examen Parcial – I TÉRMINO 2012-2013

Nombre:_____ Matrícula:_____

1. (10) Categorice las siguientes aplicaciones de acuerdo a si deben priorizar latencia o throughput: **skype, ftp, youtube, ssh/telnet/vnc, email, ping, facebook, monitoreo de sistemas**. De 2 ejemplos más de aplicaciones para cada categoría.
2. (10) De manera breve y concisa, describa el porqué de las 2 principales decisiones de diseño de los procesadores RISC.

> *It is easy to see by formal-logical methods that there exist certain [instruction sets] that are in abstract adequate to control and cause the execution of any sequence of operations … . The really decisive considerations from the present point of view, in selecting an [instruction set], are more of a practical nature: simplicity of the equipment demanded by the [instruction set], and the clarity of its application to the actually important problems together with the speed of its handling of those problems.*
>
> Burks, Goldstine, and Von Neumann, 1947

3. (15) Mediante un ejemplo en MIPS, explique porqué es más eficiente utilizar aritmética de punteros para recorrer un arreglo. Compárelo a un recorrido típico utilizando índices. Utilice métricas apropiadas para su comparación (número de instrucciones por programa).
4. (10) Si el código de su programa tiene muchas instrucciones, una instrucción de salto condicional podría no ser suficiente para realizar cualquier salto. Explique porqué una solución como la identificada en el siguiente código sirve para solucionar parcialmente el problema:

```
        beq     $s0, $s1, L1
# Si L1 se encuentra muy lejos de la línea actual, esto no servirá.

        bne     $s0, $s1, L2
        j       L1
L2:
# En este caso L1 puede encontrarse mucho más lejos.
```

5. (15) Opine acerca de la siguiente aseveración: *Escribir programas en lenguaje ensamblador siempre rendirá el mejor desempeño.*
6. (10) Describa los 3 tipos de problemas (hazards) que podemos encontrar al realizar pipelining.
7. (5) Corrija el siguiente código para evitar uno de los problemas (hazards) mencionados en la pregunta anterior:

```
lw      $t1, 0($t0)
lw      $t2, 4($t0)
add     $t3, $t1, $t2
sw      $t3, 12($t0)
lw      $t4, 8($t0)
add     $t5, $t1, $t4
sw      $t5, 16($t0)
```

8. (25) Escriba una función en MIPS para **desordenar** un arreglo. Considere que dispone de una función **random** que recibe un valor mínimo ($a0) y otro máximo ($a1) y retorna ($v0) un entero que se encuentra en dicho rango.