Sistemas Digitales

Preguntas teóricas para el parcial 1C 2024

1. Observando las instrucciones que realizan operaciones aritméticas, describir a que componentes conectaría la ALU en el datapath. Explicar detalladamente el funcionamiento del siguiente programa en assembler:

```
main:
        addi a1,x0,1
        addi a2,x0,2
        bltu a2,a1,test
        sub a3,a1,a2
        jal end
test:
        sub a3,a2,a1
end:
        nop
```

2. ¿Qué significa que la arquitectura de RISCV sea modular? ¿Qué ventajas puede tener esto?
3. ¿Cuáles son los objetivos no funcionales (o métricas de diseño) qué guían el diseño del set de instrucciones de RISCV? ¿Cómo impactan en su diseño?
4. ¿Cuáles son las reglas de nomenclatura de las instrucciones del set RISC-V? ¿Qué tipo de instrucciones tiene?
5. ¿Cuál es la ventaja de tener un registro de valor constante 0? ¿Cómo maneja las escrituras a este registro?
6. ¿Cómo resuelve la lógica de control (branching)?
7. ¿Cómo resuelve el overflow?
8. ¿Cómo resuelve los saltos incondicionales? ¿Por qué lo hace de este modo?
9. ¿Cómo resuelve la multiplicación de números enteros?
10. ¿En qué consiste la convención de llamadas? ¿Qué registros se preservan? ¿Qué debe hacer la persona que escribe código que sigue esta convención con los registros que no se preservan?
11. ¿Qué sucede si no hay suficientes registros como para pasar los parámetros de una función?
12. ¿Qué son las pseudoinstrucciones? ¿Esto es microprogramación?
13. ¿Qué son las directivas de ensamblador?
14. ¿Qué significa position independent code? ¿Qué ventaja tiene sobre el código dependiente de posición?
15. ¿A qué se llama el heap? ¿A qué se conoce como un heap overflow?
16. Describa las similitudes y diferencias entre las instrucciones de formatos B y S. Idem entre las instrucciones J y U
17. Dados dos registros mostrar como intercambiarlos sin intervenciòn de un tercero
18. Sabiendo que *a1 = 0xffffffff* Cuánto queda almacenado en a2 luego de realizar: *anndi a2,a1,0xf00*

19. ¿En qué posición dentro de la instrucción se encuentran los bits de los registros destino y origen? ¿Depende del tipo de instrucción o de la instrucción en sí? ¿Por qué fue diseñado así el formato de instrucción?
20. ¿Qué problemas puede ocasionar utilizar un registro de propósito general para el PC?
21. ¿Cómo se hace una lectura del PC?
22. RISC-V lee los datos *little - endian*, qué significa, de un ejemplo. ¿Cuándo es importante?
23. RISC-V maneja el principio de "simplicidad", en relación a esto responda:
    a) ¿Acceder a un operando en registro es más rápido que buscar el operando en memoria?
    b) A partir del inciso anterior, ¿cómo cree que impacta al rendimiento del programa y a la arquitectura la cantidad de registros disponibles?
24. ¿Cómo resuelve BGT(branch great than) con Risc-v32?
25. Ensamblar el siguiente código
    add a0, a1, a6
    bltz x1, 0x0ABC