Schwindt, Ignacio Andrés: Correcciones Práctica 10

Pautas de corrección			
Concepto	**Descripción**	**Máxima nota conceptual**	**Puntaje máximo**
C1	Implementación de las nuevas instrucciones propuestas para el repertorio del TDA 1819 (ejercicio 1).	MB	10 puntos
C2	Desarrollo del programa de prueba en el lenguaje Assembler y simulación del mismo (ejercicio 2). Cálculo de tiempos e identificación de instrucciones y etapas específicas del cauce del procesador (ejercicio 3). Capturas de pantalla incluidas.	MB	15 puntos
C3	Calidad de la presentación del informe y los archivos ".vhd" y ".asm" adjuntos.	MB	5 puntos
T	**Calificación total obtenida.**	**MB**	**30 puntos**

Ejercicio 1:

- La arquitectura del TDA 1819 no se basa en el simulador "MSX86", sino en el "MSX88". El sistema de representación binario es Ca2 (Complemento a 2) y BSS (Binario Sin Signo) para expresar valores con y sin signo respectivamente.
- ¿Cuál es el bug del FF que, según se afirma, fue resuelto por mail de cátedra?
- Las explicaciones y comentarios ofrecidos para los cambios realizados en los distintos archivos del proyecto original no resultan satisfactorias. ¿Qué papel desempeña cada uno de estos archivos en el funcionamiento del procesador? ¿En qué consisten las alteraciones efectuadas y por qué se las consideró necesarias? En particular, ¿cuáles son las acciones llevadas a cabo en la unidad de control y la ALU de la CPU durante las etapas "decode" y "execute" de las nuevas instrucciones implementadas? No era necesario describir el ciclo de instrucción completo de cada una, pero habría resultado adecuado incluir una explicación de las etapas directamente afectadas por las modificaciones implementadas.
- Los cambios correspondientes al archivo "usuario.vhd" no debían indicarse en este apartado del informe, sino en la resolución del ejercicio 2.
- Habría sido preferible definir e implementar la lógica correspondiente a las nuevas instrucciones, en particular, sus respectivos códigos de operación, preservando el orden lógico y natural propuesto en el repertorio original soportado por el procesador y posiblemente evitando la repetición de código en algunas secciones del procesador que presentan idéntico comportamiento para el mismo conjunto de instrucciones (por ejemplo, en un apartado de la ALU). Por lo tanto, "dsubi" y "xnorr" se ubicarían a continuación de "dsub" y "xori" respectivamente, estableciendo nuevos valores para todos los códigos de las instrucciones posteriores en el subconjunto modificado (instrucciones aritméticas o lógicas según el caso). Además, en el paquete "repert_cpu" resulta especialmente confuso que se declare la instrucción "xnorr" al comienzo de las instrucciones lógicas, pero se le

asigne el código de operación consecutivo de aquél definido para la última instrucción lógica del repertorio original.

- ¿Por qué motivo se define en el archivo "const_cpu.vhd" el tipo de operación "EX_XNORR", a ser ejecutado en la ALU, con un criterio distinto a aquél utilizado en los otros ficheros modificados para implementar las nuevas instrucciones? Si "xnorr" fue ubicado a continuación de "notr", entonces "EX_XNORR" debería situarse luego de "EX_NOT".

Ejercicio 2:

- Al encontrarse el valor binario "10101110" conformado por solo ocho bits, habría sido más adecuado definir su respectiva variable, "casoMenor" en este caso, con el tipo de datos ".byte" en lugar de ".hword" en el programa de prueba, manteniendo así inalterable la cadena original (sin expansión del signo).
- Podría haberse planteado una versión más eficiente del programa a fin de evitar la repetición por duplicado de las instrucciones "sh r7, finalLD(r0)" y "sh r8, finalDL(r0)".
- Es incorrecta la sintaxis al final del programa: solo debería existir una única instrucción "halt" para detener el procesador, seguida por un salto de línea y una tabulación para que el ensamblador interprete el código correctamente.
- ¿Por qué se consideró necesario reflejar los resultados almacenados en la memoria de datos de la computadora una vez finalizada la ejecución tanto secuencial como segmentada del programa? La segmentación del cauce no debería repercutir sobre el comportamiento general del programa, sino únicamente sobre su tiempo de ejecución.
- ¿Cómo debían interpretarse los resultados en memoria considerando el atributo "endianness" de la arquitectura del procesador?
- El ensamblaje del programa de prueba incluido en la entrega fallaba en el simulador debido a que en la línea 26, correspondiente a la segunda instrucción "slti", su respectivo comentario se encontraba separado de ella por tabulaciones en lugar de espacios. ¿Cómo se logró en tal caso ejecutar el programa en el procesador?

Ejercicio 3:

- Podría haberse incluido una captura de pantalla del Waveform con el progreso a través del tiempo de las señales "DoneCompUser", "DoneCpuUser" y "ReadyUser" para la ejecución secuencial y segmentada del programa, destacando en particular el instante de tiempo en el cual se activa "DoneCpuUser" y finaliza el programa.
- La resolución planteada para el inciso b es incorrecta dado que no resuelve en absoluto las consignas aquí propuestas. Se solicitaba identificar las instrucciones "dsubi" y "xnorr" del programa efectivamente ejecutadas, determinar la etapa de sus ciclos de instrucción en la cual se efectuaban sus respectivas operaciones aritmético/lógicas en la ALU y establecer la duración y los instantes de inicio y fin de dicha etapa en cada caso habilitando y deshabilitando la ejecución segmentada del programa. En realidad, para esta tarea habría sido especialmente útil el estudio de la señal "InstStatesExec", disponible en el archivo "states_machine.vhd", ubicado a su vez en la ruta "Usuario\PC\2. CPU\1. UI\1. Máquina de Estados" del proyecto "TDA_1819" original, ya que contenía información detallada de todas las instrucciones del programa ejecutadas, en particular, sus cauces segmentados y su ubicación en el programa.

Calificación obtenida		
Concepto	**Nota conceptual**	**Puntaje**
C1	B+	8 puntos
C2	R	8 puntos
C3	B+	4 puntos
T	**B**	**20 puntos**