Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Prof. Archana Chidanandan

Criteria to evaluate the Assembly Language and Machine Language specifications and the document

a. For each instruction, a clear English language description is available, describing:
   i. Functionality of the instruction.
   The function definitions clearly states what each instruction does.
   ii. Number of operands and order of operands.
   The order and number of operands are listed in the table of commands.
   iii. The instruction format.
   The instruction formats are defined in the section that shows how each type of instruction is translated to machine language.
   iv. The opcode
   The opcodes are listed next to the instruction name in the table of instructions.
   v. The machine language representation for the different types of operands.
   The machine language representation is explained in the section that shows how to convert instructions to machine language.

b. For branch instructions (conditional and unconditional),
   i. The address field is large enough to hold any target address.
   The three instructions that use immediate values to branch cannot reach any target address. They use signed 4-bit and 12-bit offsets.
   ii. If the address field is not large enough, there is a sequence of operations, that will allow the flow of control to change to the target address.
   The 'jr' instruction can jump to any address in memory after the address is loaded into a register.
   iii. The documentation indicates this sequence of operations, through an example or through a clear and concise description.
   The note that describes how 'beq' can sometimes be a pseudo-instruction lists instructions that can be substituted so any address can be reached.

c. Interrupt handling
   i. The documentation states which Interrupt-handling mechanism is being implemented(Vectored Interrupts method or Status Register method).
   The type is not directly stated, but the presence of a interrupt register and the mention of a single interrupt handler suggests it is status register based. The only way to tell where the interrupt handler is located in memory is to look at the example program. They placed it at 0x0080.
   ii. The documentation states which I/O handling mechanism is being implemented(Memory-mapped I/O, Register-mapped I/O, Special Instructions).
   The I/O is handled by special registers, accessed by special instructions.
   iii. For memory-mapped I/O, the list of reserved addresses is specified, along with a description.
   N/A
   iv. For register-mapped I/O, the list of reserved registers is specified, along with a description.

A list of registers and their functions is included.

v. For special instructions, the list of instructions is listed, with a description.
The special instructions are to access the special registers. They are described to the same level as the rest of the instructions.

d. A list of special purpose registers is provided, with a description for each register.
A list of registers and their functions is included.

e. Procedures
   i. An instruction or a short sequence of instructions exists, to transfer control to a procedure.
   Yes, 'jal', 'jr', or the branches can be used to enter a procedure.
   ii. An instruction or a short sequence of instructions exists, to return control to the calling procedure.
   Yes, 'jal' stores the return address in '$ra'.

e. In a few sentences, comment on the ease of use of the document and on the ease of programming with the Instruction Set Architecture.

Aside from the interrupt address not being directly stated, the documentation was easy to understand and described the language to the point where it was not difficult to write a program in the language.