CS250 Spring 2007
Solution[1] for Assignment 5

1: Assembly (30%)

1. The first instruction (subu) addresses an immediate value through a register, while the second instruction (sw) uses a register offset addressing mode.

2. We cannot move the first two instructions alone. If we did so, there could be the case that the program would reach the instruction `addu $sp, $sp, 8` without first executing the instruction `addu $sp, $sp, -8`. This is the case that n is not greater than 1 (and the recurs part of the code would not be executed). So, we would deallocate space that we never allocated before - this is a memory management error.

If we do want to move those two instructions, we should also move the instruction `addu $sp, $sp, 8` to be the last one before the return label. However, this approach is not recommended, as it does not follow the convention (see pp. A25-26 of James Larus,"Assemblers, Linkers, and the SPIM Simulator" at the WebCT lab resources).

3. We do not save $ra at the very beginning of the function (and neither restore it at the end). Although $ra is indeed saved when the main program executes the "jal fact" instruction, it is necessary to save it again in the fact function, because fact is a recursive function. So, the first time it calles itself (by using the jal instruction), the $ra will be overwritten and the old value will be lost, if we have not previously saved that on the stack.

2: Memory Management (25%)

The Figure located at `http://en.wikipedia.org/wiki/Image:Page_table_actions.png` may be helpful. There are more than one possible procedures that can be followed, here is one of them:

TLB and page table lookup are executed at the same time (in parallel). If there is a hit at either of those, then we know that we will find the requested data either at the cache or the main memory. The difference is that when we have a miss at the TLB (but a hit at the page table), the TLB will be updated, i.e. written, so that it includes the new address. A page table miss (not possible to have a TLB hit) either means that the page is not at the main memory and we have to access the disc, or that we have a segmentation fault. This fault happens when the virtual address does not correspond to a real address. If we access the disc, then we transfer the requested page at the main memory and update the page table and the TLB. At any case after updating the TLB and the page table, we start from the beginning: we access the TLB and we now know that we will have a hit.

We make the assumption that the page table is located at the memory and we get the following numbers:

1. (7 + 11)ns = 18ns

2. (51 + 7 + 11)ns = 69ns

[1]For any concerns, please contact the TA: Debbie Perouli

3. (51 + 7 + 11 + 51)ns = 120ns

4. (51 + 20000000 + 7 + 51)ns = 20000109ns

3: Cache (30%)

1. The missing information is how fast we can access the main memory. The lower the cost of a cache miss, the less the need for a fast cache.

2. Suppose that the memory reference needs 50ns. Then accessing the memory costs 10 times more than accessing the fast cache and 2.5 times more than accessing the slow cache. Since we can have 2MB of the fast cache for 85% hit rate, selecting the fast cache seems to be reasonable. This means that on average we will reach the data within 5ns 85% of the times and within 50ns for the rest 15% of the times.

3. We can use two levels of cash. The L1 cache will be 1MB of the fast one, while L2 cache will be 5MB of the slow cache (total $150). In this way on average we will reach the data within 5ns 80% of the times, within 20ns the remaining 18% (0.2*0.9) of the times, and within 50ns the rest 2% of the times.

4: Data Representation (15%)

1. The most significant bit is the parity bit. So, the actual value is the binary 1000001 which is the decimal 65. The ASCII character that corresponds to that value is 'A'.

2. The sequence of bits that we have received is wrong. (We require even parity, but the number of one bits in the byte is odd.) So, we cannot tell what ASCII character we should have received, as there are many different combinations of errors that may have happened.