LSU EE 4720 Homework 1 Solution Due: 10 February 2003

At the time this was assigned computer accounts and solution templates were not ready. If they become available they can be used for the solution, either way a paper submission is acceptable.

Problem 1: When compiling code to be distributed widely one should be conservative when selecting the target ISA but less caution needs to be taken with the target implementation. Explain what "conservative" and less caution mean here, and explain why conservatism and in one case less caution in the other can be taken.

Conservative means choosing an ISA variation that most users' computers implement (rather than the variation that would give best performance). Less caution means selecting an implementation that fewer people have but that would give better performance.

Conservatism is necessary for ISA selection because a computer won't run software for an incompatible ISA. On the other hand, a computer will run software compiled for a different implementation, though not as fast as if compiled for the same implementation.

Problem 2: Based on the SPECINT2000 results for the fastest Pentium and the fastest Alpha, which programs would a shameless and unfair Alpha advocate choose if the number of programs in the suite were being reduced to five. Justify your answer.

At the time this solution was written, the fastest Pentium ran at 1130 SPECint2000's and the fastest Alpha ran at 928 SPECint2000's.

The advocate would choose the five programs which performed best compared to the Pentium. They are from best to worst (of 5) mcf, vpr, bzip2, twolf, crafty. The last, crafty, is faster on the Pentium so the advocate might want just four programs. The table below shows the benchmark run times and the ratio of run times (Pentium divided by Alpha), sorted by ratio.

Benchm.	Pentium	Alpha	Ratio
mcf	231	123	1.88
vpr	210	159	1.32
bzip2	174	150	1.16
twolf	326	292	1.12
crafty	84.8	98.4	0.86
gcc	88.7	112	0.79
vortex	113	145	0.78
parser	170	256	0.66
eon	82.5	132	0.63
perlbmk	130	208	0.63
gzip	111	240	0.46
gap	75.2	164	0.46

Problem 3: The Pentium 4 can execute at a maximum rate of three instructions (actually, microops, but pretend they're instructions) per cycle (IPC), the Alpha 21264 can execute at most 4 IPC and the Itanium 2 can execute at most 6 IPC. Assume that the number of instructions for perlbmk, one of the SPECINT2000 programs, is the same for the Alpha, Itanium 2, and Pentium 4 (pretending micro-ops are instructions, if you happen to know what micro-ops are).

(a) Based upon the SPECINT2000 results (not base) for the perlbmk benchmark, which processor comes closest to executing instructions at its maximum rate? ("Its", not "the".)

The first thing to figure out is just what is meant by "**closest** to ... its maximum rate?" Another way of stating the question is: which wastes the fewest instructions?

Lets suppose that each processor was executing at its maximum rate. The total number of instructions executed would be the $\mathrm{IPC} \times \phi \times t$, where ϕ is the clock frequency and t is the execution time. According to the SPEC disclosure the Pentium 4 runs at $3066\,\mathrm{MHz}$ and takes 130 seconds to execute perlbmk and so it could execute at most 1.195740 trillion (10^{12}) instructions. Similarly, the Alpha could execute $1250\,\mathrm{MHz} \times 4\,\mathrm{inst/cycle} \times 208\,\mathrm{s} = 1.04$ trillion instructions and the Itanium2 could execute $900\,\mathrm{MHz} \times 6\,\mathrm{inst/cycle} \times 251\,\mathrm{s} = 1.3554$ trillion instructions. Since we are assuming they all execute the same number of instructions, the Alpha, which could execute the fewest instructions, comes closest to its potential.

(b) Are these numbers consistent with the expected tradeoffs for increasing clock frequency (mentioned in class) and for increasing the number of instructions that can be started per cycle?

Assume that the technology is fixed. (The transistors are not getting faster.) To increase the clock frequency we need to break instructions up into more steps, and that means fewer opportunities for overlap. That means there will be more times when one cannot find an instruction to execute (because the source operands that it needs are not yet ready). So with a higher clock frequency we would expect execution to be further from its maximum. This is true for the Pentium compared to the Alpha, but does not hold for the Itanium2.

A processor that can handle more instructions per cycle will have a more difficult time overlapping them, so one would expect it also to execute further from its maximum. This holds for the Itanium2 compared to the Alpha and Pentium 4, but does not hold for the Alpha compared to the Pentium 4.

Problem Discussion:

The analysis is based on the assumption that the same number of instructions are executed on each system, a bad assumption to make. (Sorry, I don't have the numbers.) There is also an important difference between the processors' ability to overlap instructions. The Pentium 4 and Alpha 21264 are dynamically scheduled, which means they have greater freedom in overlapping instructions. (Instructions do not have to start in program order on these processors, while they do on the statically scheduled Itanium2.) Using a technique called predication, the Itanium2 can avoid branches, an advantage the others don't have. The Itanium (VLIW) ISA has several other features designed to provide performance on modern implementations, features the older Alpha (RISC) and ancient IA-32 (maybe CISC) ISAs lack.

Problem 4: Complete the `lookup` routine below so that it counts the number of times an integer appears in an array of 32-bit integers. Register `$a0` holds the address of the first array element, `$a1` holds the number of elements in the array, and `$a2` holds the integer to look for. The return value should be written into `$v0`.

```
lookup:
        # Call Arguments
        #
        # $a0: Address of first element of array.  Array holds 32-bit integers.
        # $a1: Number of elements in array.
        # $a2: Element to count.
        #
        # Return Value
        #
        # $v0: Number of times $a2 appears in the array starting at $a0

        # [ ] Fill as many delay slots as possible.
        # [ ] Avoid using too many instructions.
        # [ ] Avoid obviously unnecessary instructions.

        # A correct solution uses 11 instructions, including 6 in
        # the loop body.  A different number of instructions can be used.

        # Solution Starts Here

       addi $v0, $0, 0
       sll $t0, $a1, 2
       add $t1, $t0, $a0
LOOP:
       beq $a0, $t1, DONE
       lw $s0,0($a0)
       bne $s0,$a2 LOOP
       addi $a0, $a0, 4
       j LOOP
       addi $v0, $v0, 1
DONE:


        # Use the two lines to return, fill the delay slot if possible.
        jr $ra
        nop
```