EGRE 426
Digital Systems
Final
Open Book/ Open Notes
12/8/08

NAME: SOLUTIONS

1. Using the simplistic assumption that all data is available as needed, and each processor is non-pipelined and can add two numbers in one unit of time. Answer the following:
   a) Using a single processor how long would it take to compute the sum of 4 numbers?
   b) Using three processors how long would it take to compute the sum of 4 numbers?
   c) Using a single processor how long would it take to compute the sum 100 numbers?
   d) Using an unlimited number of processors how long would it take to compute the sum of 100 numbers?
   e) Using three processors how long would it take to compute the sum of 100 numbers?

Answer:

a). Summering 4 numbers requires 3 additions; therefore, 3 units of time.

b). At most only two processors can be use at a time; therefore, time = $\lceil \log_2(4) \rceil = 2$

c). Summing 100 numbers requires 99 additions; therefore, 99 units of time.

d). $\lceil \log_2(100) \rceil = 7$

e). Distribute the problem across the three processors so that two of the processors sum 33 numbers and the third sums 34 numbers. The first two finish summing their 33 numbers after 32 units of time. During the next unit of time sum the results of the first two processors, and complete summing the 34 numbers on the third processor. Then sum the two results obtained at time = 33, for a total time of T = 34.

2. The program shown below uses a non-delayed "bne" instruction. Efficiently rewrite the program using a branch not equal with a delayed branch of two, "bneD2" instruction. Hint: it is not necessary to use "NOP" instructions.

```
Loop:   lw      $t0, 0($s1)
        addu    $t0,$t0,$s2
        sw      $t0,0($s1)
        addi    $s1,$s1,-4
        bne     $s1,$zero,Loop
```

Note that

```
Loop:   lw      $t0, 0($s1)
        addu    $t0,$t0,$s2
        bneD2   $s1,$zero,Loop
        sw      $t0,0($s1)
        addi    $s1,$s1,-4
```

ANS:

```
Loop:   lw      $t0, 0($s1)
        addi    $s1,$s1,-4
        bneD2   $s1,$zero,Loop
        addu    $t0,$t0,$s2
        sw      $t0,4($s1)
```

does not work since the loop is executed an extra time. To see this consider the case where $s1 = 4

3. Write the shortest possible sequence of MIPS that will perform the following operations of $7. Set the most significant byte to all 0', set the next most significant byte to all 1's, complement each of the eight bits in the next most significant byte, and leave the least significant byte unchanged. For example, if before the code is executed

$$\$7 = n_{31}n_{30}n_{29}n_{28}n_{27}n_{26}n_{25}n_{24}\_n_{23}n_{22}n_{21}n_{20}n_{19}n_{18}n_{17}n_{16}\_n_{15}n_{14}n_{13}n_{12}n_{11}n_{10}n_{9}n_{8}\_n_{7}n_{6}n_{5}n_{4}n_{3}n_{2}n_{1}n_{0}$$

After your code is executed the value in $7 should be

$$\$7 = 0000000\_11111111\_\overline{n}_{15}\overline{n}_{14}\overline{n}_{13}\overline{n}_{12}\overline{n}_{11}\overline{n}_{10}\overline{n}_{9}\overline{n}_{8}\_n_{7}n_{6}n_{5}n_{4}n_{3}n_{2}n_{1}n_{0}$$

You may use $6 as a temporary register.

```
Ans:  xori   $7,$7,0xff00
      lui    $6,0x00ff
      andi   $7,$7,0xffff
      or     $7,$6,$7
```

4. A certain MIPS computer has a separate cache for instructions and data.
(a). The instruction cache is a direct mapped cache, and contains 1024 lines. Each line contains 16 instructions. How many **address** bits are required for the tag?
Answer: 1024 lines require 10 bits for the index (2^{10} = 1024). 16 X 4 = 2^6 bytes in a line requires 6 bits for the offset. The number of bits for the tag = 32 – 10 – 6 = 16 bits.
(b). The data cache is two way set associative and can hold a maximum of 8192 = 2^{13} bytes of data. If ten address bits are required for the index, how many bits are used for the tag?
Answer: Each set contains 8192/2 = 4096 = 2^{12} bytes of data. 10 index bits implies 2^{10} lines in each set. Thus, there are $2^{12}/2^{10} = 2^2$ bytes in each line requiring 2 bits for the offset. The number of tag bits = 32 – 10 – 2 = 20.