CS 0447 - Fall 2009 - Exam 1 Sample Solutions

See the problem list for the questions.

1. a) 1D3
   b) 7E569324
   c) EF7A
   d) BC3F6

2. a) 0723
   b) 17625511444
   c) 167572
   d) 2741766

3. a) 0000 1010 1010 1111
   b) 1011 1100 0111 1001 0000 0001 1000 0011
   c) 0110 1111 1101 0010 0011 1110 1010 0101
   d) 0100 1101 1000 1000 1111 0000 0111 0010

4. First, convert into binary. Put leading zeros to make the length a divisable by 3. From (3) with leading zeros:
   a) 000 000 101 010 101 111 ==> 005257
   b) 010 111 100 011 110 010 000 000 110 000 011 ==> 27436200603
   c) 001 101 111 110 100 100 011 111 010 100 101 ==> 15764437245
   d) 001 001 101 100 010 001 111 000 001 110 010 ==> 11542170162

5. to 19. See solutions on the web.

20. a) HDTV, b) digital camera, c) dishwasher

21. Gmail, Google, my.pitt.edu

22. Control, data path, memory, input, output

23.
a) 0x012a4020
b) 0x22300064
c) 0x00031442
d) 0x11e00007
   to get the encoding for the branch label, we need to compute the amount to add to the branch's instruction address.

   recall that a branch's target address is:
   PC = PC + 4 + (sign_extend(offset) << 2)

   to get the offset, we basically do this in "reverse":
   offset = (label address - (branch address + 4)) >> 2
          = (0x0040004c - (0x0040002c + 4)) >> 2
   = (0x0040004c - 0x00400030) >> 2
   = (0x1C) >> 2
          = 7

   notice in the answer that the lower 16 bits are 0x7.

24. To allow loading 32-bit immediates.

25. nor $t0,$t0,$0

26. No answer.

27. a) 5
    b) It would be very cumbersome to quickly convert to and from this notation. It's much easier to remember a lookup table with only 16 numbers. Furthermore, most binary numbers in computer systems are in "chunks of 4", which match naturally to hexadecimal notation.

28. Translates a program written in a high-level language, like Java, C++ or C#, into assembly language.

29. Data types, registers & usage conventions, processor modes, exception handling and compatibility issues.

30. 31 Register $0 is always zero. The other general-purpose registers can be changed.

31. 0x400 (1024)

32. 0x800 (2048)

33. 0x0FFF0FFF

34. These accesses cause a misalignment error:
    access 2, access 3, access 8,

35. after the first lw $t3=0x03020100
    after the second lw $t3=0x00010203

36. first, the “little end” is 0xFF, so this is the loaded value for little-endian. next, $t3 = 0xFFFFFFFF because lb **sign extends** the value 0xFF into a 32-bit value.

37. first, the “little end” is 0xFF, so this is the loaded value for little-endian. next, $t3 = 0x000000FF because lbu **zero extends** the value 0xFF into a 32-bit value.

38. jal bart

39. jr $ra