CSSE2010 / CSSE7201 – Introduction to Computer Systems
Answers to Exercises – Week Seven
Instruction Set Architecture

Answers

Some of the questions below are taken from or based on questions in Tanenbaum, Structured Computer Organisation, 5th edition.

1. How many memory reads are required to read a word of the given width in each of the following circumstances. (If more than one answer is possible depending on the alignment, then give the best-case and the worst-case.)
   (a) 4 byte word, 8-bit data bus, natural alignment required
   4 memory reads will be required - one per byte

   (b) 4 byte word, 16-bit data bus, natural alignment required
   2 memory reads will be required – 2 bytes per read

   (c) 8 byte word, 32-bit data bus, natural alignment required
   2 memory reads will be required – 4 bytes per read

   (d) 2 byte word, 8-bit data bus, no alignment restrictions
   2 memory reads will be required – one byte for each. It doesn't matter what the alignment restrictions are when only one byte is read at a time.

   (e) 8 byte word, 16-bit data bus, no alignment restrictions
   Best case – the word begins at an address which is a multiple of 2 bytes (this includes the natural alignment case which is a multiple of 8 bytes). In this case 4 memory reads will be required.
   The other option is if the word begins at an odd address – in this case five memory reads will be required – with the first and last reads each returning one byte of the word (plus an unwanted byte) and the second through fourth reads each returning two bytes of the word.

   (f) 8 byte word, 32-bit data bus, no alignment restrictions
   Best case – the word begins at an address which is a multiple of 4 bytes (this includes the natural alignment case which is a multiple of 8 bytes). In this case 2 memory reads will be required.
   The other option is if the word begins at an address which is not a multiple of 4 – in this case 3 memory reads will be required.

   Why is it advantageous for a processor to require natural alignment?
   Because this guarantees that you'll be able to read longer words in from memory with the fewest possible number of reads.

2. For each of the following scenarios, how wide (in bits) does the address bus need to be?
   (a) A CPU that uses byte addressing (i.e. has an addressable cell size of one byte) and supports 2^{36} bytes of memory with a 4 byte wide data bus
   The number of bits in an address is the number of bits needed to represent all possible cell addresses – in this case 36, since there are 2^{36} memory cells (each of size one byte). Each memory read will return 4 bytes (cells), e.g. reading address 4, 5, 6 or 7 will return the contents of cells 4, 5, 6 and 7. The two least significant bits of the address are irrelevant, meaning that we don't need to transmit those bits to the memory. The address bus width is

therefore the number of address bits minus the number of irrelevant bits in the address. In this case, it is 36 – 2 = 34 bits.

(b) A CPU that uses byte addressing and supports 2^{32} bytes of memory with an 8 byte wide data bus

There are 2^{32} memory cells and therefore 32 bits in each cell address. The three least significant bits of the address are irrelevant ($2^3 = 8$), so the address bus width is 32 – 3 =29 bits.

(c) A CPU that uses byte addressing and supports 2^{16} bytes of memory with an 8 *bit* wide data bus

There are 2^{16} memory cells and therefore 16 bits in each cell address. None of the address bits are irrelevant – we have to send the full address to the memory each time since we only return one byte each time. The address bus width is therefore the same as the number of address bits: 16.

(d) A CPU that has an addressable cell size of 2 bytes and supports 2^{32} bytes of memory with a 4 byte wide data bus

This CPU does not use byte-addressing. The number of memory cells is NOT the same as the number of bytes in the memory. In this case, the number of memory cells is 2^{31}, so the number of bits in a cell address is 31. Each read will return 2 cells, meaning one of the address bits is irrelevant. The address bus width is therefore 31 – 1 = 30.
(Note that this is the same answer you would get if you consider the CPU to use byte addressing.)

3. Assume that a memory contains the following byte values at the given addresses:

Address (decimal)	**Data (hex)**
1028	88
1029	01
1030	AA
1031	10
1032	F0
1033	C0
1034	00
1035	00
1036	80
1037	00
1038	00
1039	2B

If you assume that this machine uses little-endian representation, what are the values (in decimal) of

(a) the two byte unsigned integer stored at address 1028?

The least significant byte is first (at address 1028), followed by the most significant byte (at address 1029). In hexadecimal, this number will be 0188. This is equal to 392 (decimal).

(b) the two byte two's complement integer stored at address 1030?

In hexadecimal, the two byte quantity at this address is 10AA. This is equal to 4266 (decimal).

(c) the four byte two's complement integer stored at address 1028?

In hexadecimal, the four byte quantity at this address is 10AA0188. This is equal to 279576968 (decimal).

(d) the ASCII character stored at address 1039?
The byte stored at this address is 2B (hexadecimal) = 43. This is the ASCII code for the plus symbol: + (Endianness is irrelevant when considering just a single byte.)

If you assume that this machine uses big-endian representation, what are the values (in decimal) of
(e) the two byte unsigned integer stored at address 1028?
In hexadecimal, this number is 8801. This is equal to 34817 (decimal).

(f) the two byte two's complement integer stored at address 1030?
In hexadecimal, the two-byte value stored at this address is AA10. This is equal to -22000 (decimal).

(g) the four byte two's complement integer stored at address 1028?
In hexadecimal, the four-byte value stored at this address is 8801AA10. This is equal to -2013156848 (decimal).

(h) the ASCII character stored at address 1039?
The byte stored at this address is 2B (hexadecimal) = 43. This is the ASCII code for the plus symbol: + (Endianness is irrelevant when considering just a single byte.)

4. Identify the type of addressing used for all of the following Atmel AVR instructions. (You may need to refer to the Instruction Set Manual or the Instruction Set reference.)
General comment – when trying to decide the type of addressing used, concentrate on operands other than the 8-byte general purpose registers and I/O register numbers. If there are only general purpose and/or I/O registers listed then it is register addressing. Otherwise, it is some other form of addressing determined by the other operand. (This is because the AVR does not use the 8-bit general purpose registers to hold addresses – this only happens with the 16-bit registers X, Y and Z. Remember though that the X, Y and Z registers are actually each made up of two 8-bit general purpose registers.)
(a) add r5,r3
Register addressing (Atmel calls this Register Direct) – the register number(s) is in the instruction.
(b) adiw ZH:ZL, 63
Immediate addressing.
(c) andi r19,$AA
Immediate addressing.
(d) mov r3,r5
Register addressing (Atmel calls this Register Direct)
(e) in r25,$16
Register addressing (Atmel calls this I/O Direct)
(f) ld r2,Y
Indirect addressing (Atmel calls this Data indirect)
(g) ldi r30,$F0
Immediate addressing
(h) lds r10,$FF00
Direct addressing. (Atmel calls this Direct Data addressing.)