Exercises 1
Assembly and Machine Languages

Computer Organization and Components / Datorteknik och komponenter (IS1500), 9 hp
Computer Hardware Engineering / Datorteknik, grundkurs (IS1200), 7.5 hp

KTH Royal Institute of Technology
Friday 18th December, 2020

Number Systems

1. Convert the following numbers into binary representation. Do it by hand, without a calculator.

   (a) $\texttt{151}_{10}$

   (b) $\texttt{3F}_{16}$

   (c) `0x19E3A7F3`

2. Convert the following numbers into hexadecimal numbers. Do it by hand, without a calculator.

   (a) $\texttt{1101 0101 0011 1111}_{2}$

   (b) $\texttt{0111 1001 1110 1001}_{2}$

   (c) $\texttt{151}_{10}$

3. Assume that the following numbers are encoded as 12-bit signed values in two's complement form. Sign extend *and* zero extend them into 16-bit values. Do it by hand and answer in hexadecimal form.

   (a) $\texttt{-59}_{10}$

   (b) $\texttt{-397}_{10}$

   (c) $\texttt{638}_{10}$

4. For the 32-bit value `0x327F9521`, state what the most significant byte (MSB) and the least significant byte (LSB) are. Also, explain the terms *byte*, *word*, and *nibble*.

Assembly Languages

5. Construct a MIPS assembly function that copies data between two memory addresses. The first function parameter is the source address, the second parameter the destination address, and the third parameter the size of the data that should be copied (in bytes). The function must copy one byte each time, that is, use the load byte and the store byte instructions. The function must also follow MIPS register calling conventions.

6. Assume that store byte, store word, load byte, and load word all take the same number of clock cycles. Create a new memory copy function that is more efficient than the previous function.

7. Consider the following C program:

```
int factlist[8];

int fact(int n){
  int r = 1;
  while(n > 0){
    r = r * n;
    n--;
  }
  return r;
}


void makelist(int start, int length){
  for(int i=0; i<length; i++){
    factlist[i] = fact(start);
    start++;
  }
}

int main(){
  makelist(3,sizeof(factlist)/sizeof(int));
  return 0;
}
```

   (a) Explain what the C program is doing.

   (b) Translate the C program into a MIPS assembly program, with the following constraints:

      - All functions must follow MIPS calling conventions for how to handle registers. This means that save registers (e.g., `$s1`) and temporary registers (e.g., `$t1`) must be handled correctly.
      - The processor is assumed to not have branch delay slots.
      - Make use of PUSH and POP macros, as defined in Lecture 3.
      - The program needs to have correct assembly directives, including handling of data and code.

   (c) Assume that function `makelist` should be possible to be called from other C or assembly files. In such a case, how should then the assembly program be changed?

(d) Assume now that the processor has branch delay slots. Update the assembly program from the previous exercise so that it supports branch delay slots. Try to make the program as efficient as possible. You may save data on the stack in another way than using the PUSH and POP macros.

Machine Languages

8. Assume that the following MIPS assembly code is entered and assembled.

```
lui    $t0,0xffff
li     $t1,0x123489ab
sw     $t1,4($t0)
```

(a) What kind of instruction is `li`? How is it encoded using basic instructions?

(b) Show a memory dump of the relevant memory area after that the `sw` instruction has been executed. Assume a big-endian processor.

(c) Show the memory dump again, assuming a little-endian processor.

9. Consider the following MIPS code.

```
        add     $s1,$s2,$s3
loop:
        addi    $t2,$t2,-4
        sb      $s1,-78($t5)
        bne     $t2,$0,loop
```

(a) Create the machine code encoding for the above instructions. Answer as a list of 32-bit hexadecimal values.

(b) Represent the encoded `addi`-instruction as a C expression, using bitwise OR and shift operators.