Computer Systems and -architecture

Assignments: Resit

1 Ba INF 2016-2017

Bart Meyers
bart.meyers@uantwerpen.be

Stephen Pauwels
stephen.pauwels@uantwerpen.be

Time Schedule

Exercises are made individually. Put all your files in a tgz archive, as explained on the course's website, and submit your solution to the exercises on Blackboard. **Write down all intermediate results on how you obtained the results.**

- Deadline: **August 31, 23u55**

Data Representation

1. Convert these positive numbers to base 10.
   (a) $(1110001110)_2$
   (b) $(14A6)_{16}$
   (c) $(10110)_2$
   (d) $(1246)_8$
2. Convert to base 10.
   (a) $(11001001)_2$ (2's complement)
   (b) $(11)_2$ (2's complement)
   (c) $(.123)_4$
3. Convert to base 2.
   (a) $(666)_{10}$
   (b) $(9876)_{10}$
   (c) $(32AD)_{16}$
   (d) $(3.14)_{10}$
   (e) $(EA)_{16}$
4. Convert to base 2. Represent the negative numbers with 8 bits in *signed magnitude*, *one's complement*, *two's complement* and *excess 128*.
   (a) $(-96)_{10}$
   (b) $(-1)_{10}$
   (c) $(-69)_{10}$

(d) $(-15)_{\mathbf{16}}$

5. For the following single-precision IEEE 754 bit patterns, show the numerical value as a base 2 significand with an exponent (e.g. $+1.11 \cdot 2^5$).

   (a) `0 10101011 10111010000000000000000`

   (b) `1 10000000 10010000000000000000000`

   (c) `0 00000000 11110100000000000000000`

   (d) `0 11111111 11010000010010010010000`

   (e) `1 00001110 11100110000000000000000`

   (f) `1 11111111 00000000000000000000000`

   (g) `0 00001011 11101100000000000000000`

6. Represent these numbers in the *IEEE-754 (single precision)* format.

   (a) $(2314.25)_{10}$

   (b) $(2017)_{10}$

   (c) $(12.123)_{10}$

   (d) $(11.11 * 2^{-131})_2$ (denormalized)

   (e) $+\infty$

   (f) $+0$

7. Suppose we are using a 22 bit floating point, in a normalized, base 16 floating point format, with a sign bit, followed by a 5-bit exponent with a certain bias, followed by four base 16 digits.

   (a) Determine the bias we have to use for the exponent, assuming we do not want to change the range of exponents we would have reached when using a 5-bit 2's complement exponent.

   (b) Represent the number $-0.1775 * 10^2$ in our new format (with the bias from the previous question) as a binary string.

   (c) What is the largest possible error that can be made using this representation?

   (d) What is the smallest gap using this representation?

MIPS

1. Convert the C++ code below to a MIPS program.

```
int i  = 0;
while (i < 11)
{
   cout << i++ << endl;
}
```

2. (a) Write a MIPS program that reads an integer n (using a syscall), after which it reads n integers (using syscalls), and stores them in an array. Because you don't know the size of the array in advance, you will have to allocate space for it on the heap (*Hint: use syscall 9 for* `sbrk`).

(b) Add a subroutine that prints an array. The subroutine has two parameters: the address of the first element of the array and the number of elements in the array. Call the subroutine with the array on the heap. Use a stack frame (or activation record) in your implementation!

(c) Add a subroutine that sorts an array. Call the subroutine with the array on the heap. Implement the algorithm from the C++ function below. Use a stack frame (or activation record) in your implementation!

```
void sort(int array[], int arrayLength)
{
   int nrOfSwaps;
   do {
      nrOfSwaps = 0;
      for(int i = 0; i < arrayLength - 1; i++) {
         if( a[i] > a[i + 1] ) {
            int temp = a[i];
            a[i] = a[i + 1];
            a[i + 1] = temp;
            nrOfSwaps++;
         }
      }
   }
   while(nrOfSwaps > 0);
}
```

(d) Now add a recursive procedure that sorts an array of integers using the quicksort algorithm. Call the procedure with the array on the heap, and left = 0, right = array size.

```
void quickSort(int arr[], int left, int right) {
        int i = left, j = right;
        int tmp;
        int pivot = arr[(left + right) / 2];

        /* partition */
        while (i <= j) {
                while (arr[i] < pivot)
                        i++;
                        while (arr[j] > pivot)
                                j--;
                        if (i <= j) {
                        tmp = arr[i];
                        arr[i] = arr[j];
                        arr[j] = tmp;
                        i++;
                        j--;
                }
        };

        /* recursion */
```

```
        if (left < j)
                quickSort(arr, left, j);
        if (i < right)
                quickSort(arr, i, right);
}
```

3. Write a MIPS program that prints out the English word for an entered number. If the user enters "2", the output is "two". The program gives output for each number from 0 to 4.

   (a) Use if-then-else operations to implement this.

   (b) Use a case table with indirect addressing to implement this. *Hint: use* `la` *and* `jr`*.*

   Make a comparison of these two solutions in terms of (a) number of instructions used (size of the program) and (b) number of instructions executed (performance of the program). Compare the worst case and best case scenarios (with respect to the input value) of the two implementations. Give your conclusion.