My lab machine is pc01 (jia@pc01)

Task 1

The parameters of the function are res, base, expo, and mod. The implemented function has additional variables which indicate certain characteristics of the given parameters. These are indicated by the first letter (X) of the parameter and then the characteristic (Xp, Xsize, Xsign, Xsec, Xp_marker). The parameter res will contain the result of the calculation, mod contains the modulus, base contains the base value of the exponentiation, and expo contains the exponent ($RES = BASE^{EXP} mod MOD$). Furtheremore, W indicates the window size of the implemented left to right sliding window exponentiation algorithm

The algorithm used is a modified version of the left-to-right k-ary exponentiation - left-to-right sliding window modular exponentiation. The implemented function contains several steps:

1. Initialization: Variables are initialized, memory is allocated.
2. Precomputation: The values for the array with precomputated values ($BASE^{2 \cdot i+1}$, smallest odd powers of BASE within window size) are computed (and $BASE^2$). The amount of precomputation depends on the window size W.
3. Main Loop: iterate over the bits of the exponent from $esize - 1$ to 0 ("left-to-right") and apply squares and multiplications respectively. Thereby, the leftmost unproceeded bit is searched and shifted into position, so that the precomputed values can be multiplied (precomputed values are temporarily stored in the variable base_u).
4. Finalize: the modulo value is shifted at the beginning depending on the amount of leading zeroes in the modulo value. At the end, this is being reverted, the code makes sure that the result is put in res, possibly negative results are handled and memory is freed.

Task 2

The conversion from probe times to a sequence of S and M is achieved by a simple iteration over all measurements and just accepting the first occurence of a cache hit of each operation. To find the exponents d_p and d_q we have to search for a break and/or for the occurence of multiple MUL operations. By ignoring the blocks of MUL operations at the beginning and end of the measurement we can detect two blocks in which MUL operations only happen after SQR operations. Furthermore, by looking into the code, we can see the precomputation in the mul_powm function. For a window size of $W = 2$ we have one multiplication for $BASE^2$ and one for $precomp[1] = BASE^3$ which makes exactly two MUL operations at the beginning of a block of measurements of an exponentiation. By observing such an (timing) behavior we can identify the exponent blocks.

examplary console output:

Task 3

The retrieved sequences from out-01.txt can be found in task3.txt

The correction of measurements depends on the occuring problems. Here are several heuristics which I think can improve the signal quality (the signal of my traces was not good enough and the provided traces had too much variance for my attempts):

- probes with much noise can be corrected by applying windows in which values are taken, e.g. when there are cache hits within two measurements we can just count the first one and demand e.g. at least three cache misses until a cache hit is counted for the respective operation
- duplicate positives of the MUL operation can be corrected by either adding the missing SQR operation or removing one of the MUL operations. Since we just missed a SQR operation very likely, we should preferably add the SQR operation which appears to be missing
- use multiple measurements: we can measure multiple times, extract the exponents d_p and d_q respectively and count occurences of operations at each position. By doing it a lot of times, the real exponents should be observable. Since SQR occurs much more often than MUL, the occurence of MUL and SQR operations need additional weighting. By experimental attempts weights which depend on just how often operations occured does not seem to be sufficient.

Task 4

I do not know why my code takes so long, so I could not check if my resulting values were correct. Since the provided traces also differ very much from each other and my code depends on perfect sequences of S and M, I'll provide the code and describe what I tried:

- parsing of measurements:

  first, the cache timing measurements need to be transcribed to sequences of S and M. Thereby, we need to identify the two blocks in which d_p and d_q were processed by time (in cycles/time of measurement) and pattern of operations (multiple MUL operations).

  Then we have to identify the 0s, 1s, and unknown (marked with question marks) bits of $d_i, i \in \{p, q\}$.

- retrieve d_p, d_q, p, and q:

  We then try to retrieve d_p, d_q, p, and q bitwise by following idea:

  We know modulus N, public exponent e, many bits of d_p and d_q. We know that the LSB of p and q must be 1 since they are primes. We will use following equations

  $$p \cdot q = N, e \cdot d_p = k_p \cdot (p-1) + 1, e \cdot d_q = k_q \cdot (q-1) + 1$$

  We use these equations to show that the LSB of d_p and d_q are 1 too. If we only look at the LSB, we can see, that if $LSB(i) = 1, i \in \{p, q\}$ we have $LSB(e) \cdot LSB(d_i) = k_i \cdot 0 + 1$ and since $LSB(e) = 1$ it follows that $LSB(d_i) = 1, i \in \{p, q\}$!

  Since $(k_p - 1) \cdot (k_q - 1) = k_p k_q N \mod e, 0 < k_p, k_q < e \Leftrightarrow k_q \equiv (1 - k_p) \cdot (k_p \cdot N - k_p + 1)^{-1} \mod e$ and e known and relatively small we can compute all combinations for $k_i, i \in \{p, q\}$. We try to compute d_p, d_q, p, and q bit by bit with the following procedure (all steps at every bit position):

  1. test valid combinations of candidates for p and q for the equation $p \cdot q = N$. Since we only have two new bit positions for p and q we have maximum 4 combinations.
  2. test valid combinations of candidates for d_p and k_p for the equation $e \cdot d_p = k_p \cdot (p-1) + 1$. We computed combinations for k_p and have a one or two candidates for p. Since we recovered several bits of d_p we can exclude multiple options which reduces complexity.
  3. validate k_p by using the respective values for k_q in the equation $e \cdot d_q = k_q \dot{(} q - 1) + 1$ for all possible combinations of d_q and q. k_p and k_q are dependant of each other, p and q also.