9.1 WCET Analysis

Determine the WCET of the following program which is written in pseudo code.

```plaintext
a = 1;
if (b >= 0) then {
  i = a + 2;
  n = (a + 3)^2 + 2;
  m = a + 3;
  b = 6;
} else {
  l = a + 6;
  n = l + 8;
  i = a + 4;
}
while (n >= i) do {
  if (b > 5) {
    a = 3;
    l = l + 1;
  } else {
    m = m + 1;
    a = 4;
  }
  i = i + 1;
}
```

The underlying machine executing the program is specified as follows:

- the processor has no pipeline
- the processor has no registers, i.e. all variables are stored in memory
• in an assignment, predicate evaluation, or arithmetic operation, data variables are accessed in order from right to left e.g.

\[ a = b + c \]

variables are accessed in the order c, b, a

• consider only the data cache

• reads and writes from/to a data variable are treated the same way. If the required data is in the cache, there is a cache HIT and it takes 1 cycle to read/write the data. If the data is not in the cache, there is a cache MISS and it takes 100 cycles to read/write the data

• execution model is very simplified. No temporary results are considered. Execution times of all operations (logical, mathematical, branching, and their combinations) on the processor are 1 cycle e.g.

\[ a = b \times c + d / v^2; \]

takes 1 cycle to execute, NOT considering the loading and storing of data variables (the WCET of a statement will mainly depend on cache MISS and cache HIT of the data variables participating in the statement). Assignments such as a=2 depend entirely on whether the variable a is in cache or not.

• the data cache has only one cache set, i.e. it is fully associative. It can store 4 blocks of data. Each cache block can contain exactly one data variable

• the cache uses LRU replacement policy

Initial conditions: \( a, b, i, l, m, n \in (-\infty, \infty) \) and cache state is \((b, -, -, -), \) i.e. b is the most recently used variable and all other blocks are empty.

Tasks:

(a) Determine the basic blocks of the program

(b) Determine the intervals for each variable at each program statement using static value analysis

(c) Determine the WCET for each basic block

(d) Write down the ILP whose solution would determine the WCET of the whole program