7. Memory Latency Tolerance [60 points]

Assume an in-order processor that employs runahead execution, with the following specifications:

- The processor enters Runahead mode when there is a cache miss.
- There is a 64KB cache. The cache block size is 64 Bytes.
- The cache is 2-way set associative and uses the LRU replacement policy.
- A cache hit is serviced instantaneously.
- A cache miss is serviced after X cycles.
- The cache replacement policy chooses to evict a cache block serviced by Runahead requests over non-runahead requests. The processor does not evict the request that triggers Runahead mode until after Runahead mode is over.
- The victim for cache eviction is picked at the same time a cache miss occurs.
- Whenever there is a cache miss, the processor always generates a new cache request and enters Runahead mode.
- There is no penalty for entering and leaving Runahead mode.
- ALU instructions and Branch instructions take one cycle each and never stall the pipeline.

Consider the following program. Each element of array A is one byte.

```
for(int i=0;i<100;i++) \\ 2 ALU instructions and 1 branch instruction
{
  int m = A[i*32*1024]+1; \\ 1 memory instruction followed by 1 ALU instruction
  26 ALU instructions
}
```

(a) After running this program, you find that there are 50 cache misses. What are all the possible values of X?

$30 < X < 61$.

(b) Is it possible that every cache access in the program misses in the cache? If so, what is the value of X that will make all cache accesses in the program miss in the cache? If not, why? Show your work.

A cache latency of less than 31 cycles will not generate any cache requests during the runahead mode. A cache latency of at least 61 cycles will generate two fetches to the cache. The second fetch will evict the first runahead request in the cache, causing every access into a cache miss.

(c) What is the minimum number of cache misses that this program can achieve? Show your work.

50 misses.

(d) Assume that each ALU instruction consumes 1uJ, a cache hit consumes 10uJ, and a cache miss consumes Y uJ. Does there exist a combination of X and Y such that the dynamic energy consumption of Runahead execution is better than a processor without Runahead execution? Show your work.

No. With Runahead execution, the processor will always have to fetch extra instructions. With runahead execution, the processor will cause at least 100 cache misses (some in Runahead mode) and 30*100 ALU instruction.

(e) Assume the energy parameters in part d. What is the dynamic energy consumption of the processor with Runahead execution in terms of X and Y when X generates the **minimum** number of cache misses? Show your work.

At most, this program will convert 50 cache misses to cache hits while leaving the other 50 be cache misses. This case will happen if and only if the cache latency (X) is between $31 < X < 60$
With runahead: $misses * Y + ALU * 1 + hits * 10$+3 instructions at the end of the loop+100∗ALU in Runahead mode
With runahead: $3 + 30 * 1 * 100 + 50 * 10 + 100 * Y + 100X = 3503 + 100Y + 100(X - 1)$

(f) Assume the energy parameters in part d. What is the dynamic energy consumption of the processor with Runahead execution in terms of X and Y when X generates the **maximum** number of cache misses? Show your work.

In the worst case, every cache accesses is a miss. This will happen when $X <= 30$ and $X > 60$ However, in this case, we have to break the calculation for the energy consumption into two cases, the first case is when $X < 31$ and the second case is when Runahead execution generates extra cache requests.
When $X < 31$, the energy consumption is $3003 + 100Y + 100X$.
When $X > 60$, there will be $\lfloor X/30 \rfloor$ additional memory access in Runahead mode. So, the energy consumption is $3003 + 100(X - \lfloor X/30 \rfloor) + 100Y * \lfloor X/30 \rfloor$.