國立成功大學

110學年度碩士班招生考試試題

編　號：200

系　所：電機資訊學院-資訊聯招

科　目：計算機組織與系統

日　期：0202

節　次：第 1 節

備　註：不可使用計算機

編號： 200 國立成功大學 110 學年度碩士班招生考試試題

系　　所：電機資訊學院-資訊聯招

考試科目：計算機組織與系統　　考試日期：0202，節次：1

※考生請注意：**共兩大題**，本試題不可使用計算機。請於答案卷(卡)作答，於本試題紙上作答者，不予計分。

1. [50%] Convolution neural networks (CNNs) have been widely adopted in many different application domains. Nevertheless, as CNNs incur high computation overhead and large memory footprint, it is an important task to customize the designs of embedded processors so as to meet the timing constrains required by target applications. Among other types of computations, matrix operations are time-consuming functions in CNNs and are important targets for hardware and software optimizations to facilitate the CNN computations on such platforms. Given the partial C code for the convolution operations, there are three N-by-N matrices (A, B, and C), and they keep double-precision floating-point numbers (each number is eight bytes). Please answer the following questions related to processor designs and software optimizations.

```
for (int x = 0; x < N; ++x)
  for (int y = 0; y < N; ++y)
    for (int z = 0; z < N; z++)
      C[x+y*N] += A[x+z*N] * B[z+y*N];          (a)
```

   (1) [5%] Please write down the MIPS assembly code for the C statement (a). NOTE: Assume that the variables, C[x+y*N], A[x+z*N], B[z+y*N], have already been loaded into the CPU registers, $f4, $f6, and $f8, respectively.

   (2) [5%] Which datapath(s) of the target CPU should be optimized if we want to accelerate the computation speed of above operations?

   (3) [5%] Sometimes, embedded processors may not have the hardware support of floating-point operations. On such embedded platforms, the floating-point operations could be emulated by a software implementation of the floating-point operations (as library routines) or by integer operations. What is the technique of the floating-point emulation with integer operations?

   (4) [5%] Continue with the above question. Which one of the two emulation techniques is better for computation efficiency?

   (5) [10%] On a single-core embedded processor, which type of instructions is often supported by the embedded processor to take advantage of the data-level parallelism found in the above example code? Please rewrite the above code into the version with such type of instructions.

   (6) [5%] When N is 64, what is the smallest L1 data cache size, which can accommodate the data of all three matrices and provide no cache conflicts? NOTE: The cache should be sized in powers of 2.

   (7) [5%] Usually, to ensure the matrix elements can fit in the cache enjoying spatial and temporal locality, it is a common practice to rewrite the code into the blocked version so that the computations are done within submatrices. Please determine the maximum matrix size, M, so that the three submatrices can fit in a 16KB data cache.

(8) [10%] Continue with the above question. Please rewrite the above C code into the ***blocked*** version with the calculated M.

2. [50%] Consider a *key-value* (*KV*) storage. A KV pair stored in the storage consists of a constant length of an alphabetical string as a key, and its associated value is simply a variable-length bit string. The KV storage provides `PUT(x,v)` and `v=GET(x)` APIs to applications, where `PUT(x,v)` stores a key-value pair in the storage with the designated key `x` and value `v`, and `GET(x)` returns the value `v`, given the queried key `x`, previously stored in the KV store. Additionally, `SCAN(x1,x2)` offered by the KV storage returns a list of KV pairs such that the keys of the retrieved data items are no less than `x1` and no greater than `x2`. Let the KV storage be operated in a computer system equipped with sizes of x volatile memory space and of y persistent storage, where $x \ll y$.

Notably, applications may introduce two types of workload patterns to the KV storage. The *random access* pattern denotes keys generated by `GET(.)`, `PUT(.)` and `SCAN(.)` are random, while the *sequential access* represents the scenario that data items are addressed with increasing (or decreasing) keys. Secondly, `GET(.)`, `PUT(.)` and `SCAN(.)` operations can be performed concurrently on the fly by the KV store. Thirdly, the total storage size of KV pairs stored in the system may be greater than x, the available volatile memory space as aforementioned. Finally, the KV storage may fail anytime and support the recovery for committed `PUT(.)` operations.

Please discuss designs and implementations of operating systems to provide such a KV storage service, considering to optimize the performance metrics in terms of delays and throughputs in manipulating `GET(.)`, `PUT(.)` and `SCAN(.)` operations. You shall state clearly subject to the following techniques in order.

(1) [5%] In-memory indexing
(2) [5%] In-disk indexing
(3) [5%] Caching
(4) [5%] Paging
(5) [5%] Batched I/O
(6) [5%] Multithreading and thread scheduling
(7) [5%] Shared memory and consistency
(8) [5%] Encoding/Decoding
(9) [5%] File system block layout
(10) [5%] File system compaction