編號： 213　　**國立成功大學 103 學年度碩士班招生考試試題**　　共 6 頁，第 1 頁

系所組別：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：0222，節次：1

※ 考生請注意：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. [10%] Explain the following term in English. For each term, please use less than 100 words.
   (a) [2%] Power Wall
   (b) [2%] GPGPU
   (c) [2%] Restartable instruction
   (d) [2%] Write-allocate policy
   (e) [2%] NUMA

2. [10%] Suppose we have developed new versions of a processor with the following characteristics.

Version	Voltage	Clock rate
Version 1	1.2 V	810 MHz
Version 2	1 V	1 GHz

   (a) [5%] How much has the dynamic power been reduced if the capacitive load does not change?
   (b) [5%] Assuming that the capacitive load of version 2 is 80% the capacitive load of version 1, find the voltage for version 2 if the dynamic power of version 2 is reduced by 20% from version 1.

3. [5%] Translate the following C code to MIPS instructions. Assume that the variables *c* and *d* are assigned to registers *$s0* and *$s1*, respectively. Assume that the base address of the arrays *A* and *B* are in registers $s6 and $s7, respectively.

c = d – A[B[2]];

4. [5%] What decimal number does the following bit pattern represent if it is a floating point number? Use the IEEE 754 standard.

101011111011010000000000

5. [20%] Assume an instruction pipeline for a high-speed, load/store processor with the following instruction classes

ALU	ALUop	Rdst, Rsrc1, Rsrc2
ALUimmediate	ALUiop	Rdst, Rsrc1, imm
Load	MEMop	Rdst, n(Rsrc)
Store	Memop	n(Rsrc2), Rsrc1
Conditional branch	BRop	Rsrc1, Rsrc2, offset
Jumps	JMP	Rdst

Each instruction takes one machine word. The only memory-addressing mode supported is base register plus a signed offset. Conditional branches compare the two branch source operand values using the ALU. The branch target address is computed on a separate address generation adder contained in the control unit of the machine. Register file writes occur in the first half of a cycle and register file reads occur in the second half.

(背面尚有題目，請繼續作答)

※ 考生請注意：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。

The machine uses virtual memory address, with separate instruction and data TLBs. It also has a physical addressed (tagged) direct mapped L1 Icache and a physically addressed set associate L1 Dcache (Accesses that miss in the L1 cache cause the instruction pipeline to stall.) The ALU used during the EX cycle is pipelined and takes one cycle to complete an addition/subtraction/logical operations and two cycles to complete a multiplication. The time taken by the key components (that already includes the pipeline register write time, the interconnect delay, and any necessary multiplexors) is as follows:

I o D TLB access	1 ns
Icache access	2 ns
Instruction decode	1 ns
Address adder	1 ns
Dcache access	2 ns
ALU pipe stage	2 ns
RegFile read/write	2 ns per 2 port access (e.g. a 2-read and 2-write port RegFile takes 2 ns, a 4- read-4 write port RegFile takes 4 ns, etc.)

Answer the following questions about your pipeline.

(a) [7%] Draw the **shortest** possible instruction pipeline (i.e. the pipeline with the fewest stages) while ensuring that there are no structural hazards. For your pipeline, give a name for each stage along with a short description of what activities occur during that stage.

(b) [3%] What is its clock rate?

(c) [5%] Give a MIPS instruction example of two different data hazards that **can be** solved by forwarding (both data hazards should be different in that the forwarding is handled from different pipeline register stages). For each, explain which data are forwarded.

(d) [5%] Give a MIPS instruction example of two different data hazards that **cannot be** solved by forwarding. For each, indicate how many stalls are incurred before the hazard is resolved.

6. [10%] Consider the following page reference string:

$\{1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6\}$.

(a) [5%] In the *LRU page-replacement algorithm,* how many page faults would occur when we have 'one', 'two', 'three' page frames? Please indicate the number of page faults individually.
Assume that all frames are initially empty, and therefore your first access to each unique page will cause one fault.

(b) [5%] In the *optimal page-replacement algorithm,* how many page faults would occur when we have 'one', 'two', 'three' page frames? Please indicate the number of page faults individually.
Assume that all frames are initially empty, and therefore your first access to each unique page will cause one fault.

※ 考生請注意：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。

7. [10%] Choose the correct answers for each of the following 5 questions.

(a) [2%] The Banker's Algorithm is proposed to support the technique of:

(1) deadlock detection
(2) deadlock recovery
(3) deadlock avoidance
(4) deadlock identification

(b) [2%] For two processes accessing a shared variable, Peterson's algorithm must satisfy:

(1) semaphores
(2) mutual exclusion
(3) synchronization
(4) progress
(5) bounded-waiting

(c) [2%] Indefinite blocking may occur if we add and remove processes from the list associated with a semaphore in ________ order.

(1) first in, first out
(2) first in, last out
(3) last in, first out
(4) last in, last out

(d) [2%] A CPU-bound process _________.

(1) infrequently requests I/O operations and spends more of its time performing computational work
(2) frequently requests I/O operations and spends more of its time performing computational work
(3) infrequently requests I/O operations and spends less of its time performing computational work
(4) frequently requests I/O operations and spends less of its time performing computational work

(e) [2%] ________ increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

(1) Multithreading
(2) CPU scheduling
(3) Swapping
(4) Multiprogramming

8. [15%] To create a new process from processes, the multitasking operating systems such as Unix-like systems provide a "fork" system call. The fork function will create a copy of itself to a child process, and then other programs are executed in the child process.

(a) [5%] In the following example with the fork function, how many lines of "NCKU" will be shown in the console?

```
int main(int agrc, char **argv)
{
  int i;
  for (i=0;i<5;i++)
  {
    fork();
  }
  printf("NCKU\n");
  return 0;
}
```

(b) [5%] In the following example with the fork function, how many lines of "NCKU" will be shown in the console?

```
int main(int agrc, char **argv)
{
  int i;
  for (i=0;i<10;i++)
  {
    printf("NCKU\n");
    fork();
  }
  return 0;
}
```

(c) [5%] In the following example with the fork function, what content will be printed in the console?

```
int main(int agrc, char **argv)
{
   int x=0, i=0;
   for(i=0;i<3;i++)
   {
      fork();
      x=x+5;
   }
   printf("x = %d\n", x);
   return 0;
}
```

※ 考生請注意：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。

9. [15%] The following implementation in Java is created for a specific system utility.

(a) [7%] Could you figure out what is the system utility that can be provided from 'Manager' class? Please also simply discuss its corresponding methodology.

(b) [8%] Consider 'Manager' is used in the multi-threading mode. Is the implementation logically risky? If it is risky, please clearly point out those incorrect methods or lines, and also revise the code (written in Java). If it is not risky, please clarify your answer.

```
public interface ObjA
{
        /*check if the object is expired*/
        public boolean isExpired();
        /*return the key in the string type*/
        public String getIdentifier();
}
```

(背面尚有題目，請繼續作答)

```
public class Manager {
    private static java.util.HashMap map = new java.util.HashMap();
    private static Object lock = new Object(); /* This object ‘lock’ can act as a semaphore */

    static { /* The following code will create a background thread invoked periodically */
        try {
            Thread th_1 = new Thread(new Runnable() {
                int sleepTime = 5000;
                public void run() {
                    try {
                        while (true) { /* Sets up an infinite loop */
                            /* An iterator is used to look through all objects */
                            java.util.Iterator keys = map.keySet().iterator();
                            while (keys.hasNext()) {
                                /* Retrieve each individual key */
                                String key = (String) keys.next();
                                ObjA value = (ObjA) map.get(key);
                                if (value.isExpired()) {
                                    map.remove(key);
                                }
                            }
                            Thread.sleep(sleepTime); /* thread will sleep 5000 ms*/
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                } /* End of the run method */
            }); /* End of the thread code */

            th_1.start();   /* Starts the thread */
        } catch (Exception e) {
            e.printStackTrace();
        }
    } /* End static block */

    /* put object into the map */
    public static void putObjA(ObjA object) {
        map.put(object.getIdentifier(), object);
    }
    /* return the object if the key is found in the map */
    public static ObjA getObjA(String identifier) {
        ObjA obj = null;
        /* ‘synchronized(lock)’ can ensure that no more than one thread can lock the object
           ‘lock’ simultaneously */
        synchronized (lock) {
            obj = (ObjA) map.get(identifier);
        }
        if (obj == null)
            return null;
        if (obj.isExpired()) {
            map.remove(identifier);
            return null;
        } else {
            return obj;
        }
    }
}
```