Universiti Tun Hussein Onn Malaysia

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FINAL EXAMINATION
SEMESTER I
SESSION 2015/2016

COURSE NAME	:	COMPUTER SYSTEMS ENGINEERING
COURSE CODE	:	BEC 41603
PROGRAMME	:	BACHELOR OF ELECTRONIC ENGINEERING WITH HONOURS
EXAMINATION DATE	:	DECEMBER 2015 / JANUARY 2016
DURATION	:	2 HOURS 30 MINUTES
INSTRUCTION	:	ANSWER **ALL** QUESTIONS

THIS QUESTION PAPER CONSISTS OF **SIX (6)** PAGES

Q1 (a) Intel 80X86 processor and IBM PowerPC processor have ADD instruction in their Instruction Set Architecture (ISA). Determine whether ADD instruction for one processor can be executed by the other processor. Give reasons to support your answer.

(5 marks)

(b) Briefly explain the issue of bus architecture for multicore processor architecture in terms of long communication delay due to arbitration.

(4 marks)

(c) Sketch a diagram and explain the working principle of wormhole routing in Network on Chip (NoC) architecture.

(5 marks)

(d) Determine the network diameter for the Network on Chip (NoC) topology (i), (ii), (iii), and (iv) that are shown in **Figure Q1**.

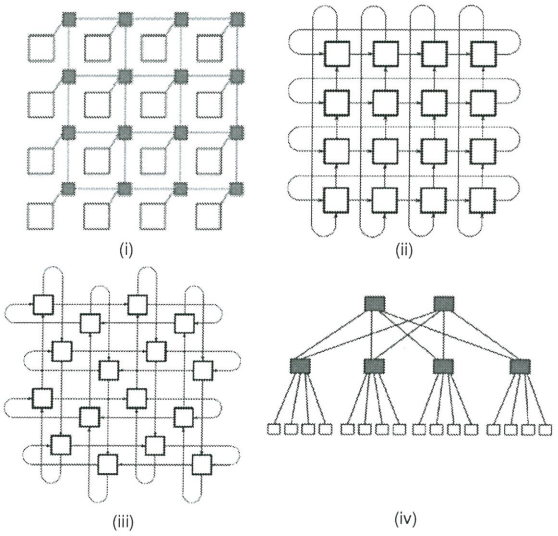

FIGURE Q1

(8 marks)

Q2 (a) Explain the concept of cache incoherency using an appropriate diagram.

(4 marks)

(b) By referring to **Figure Q2(b)**, relate the memory access mechanism to the types of memory architecture in multicore.

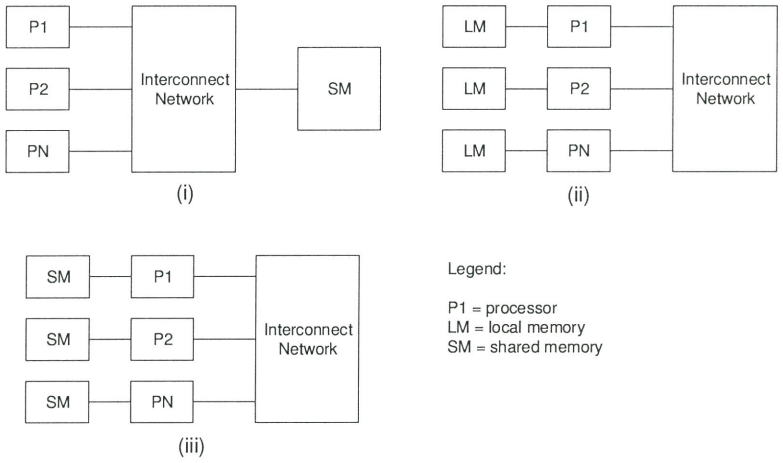

FIGURE Q2(b)

(6 marks)

(c) Predict the states of memory location 0x0FD4 in cache A, 0x0430 in cache B, and 0x03DC in cache C from **Figure Q2(c)** based on the following conditions using MESI coherency protocol.

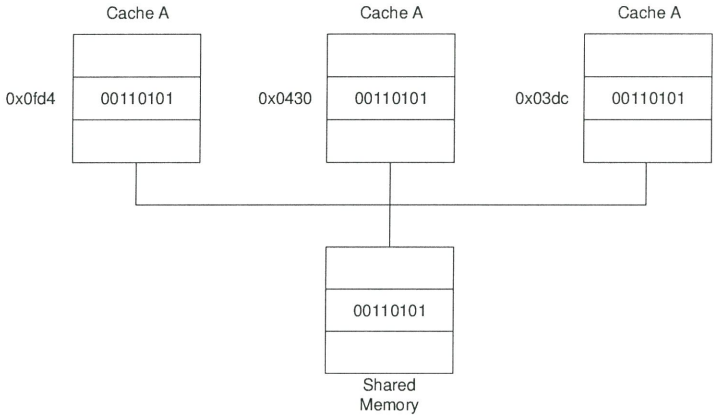

FIGURE Q2(c)

Conditions:

Initially: (1) 0x0FD4 = 00110101, 0x0430 = 00110101, 0x03DC = 00110101
Then, (2) processor A modifies cache A address 0x0FD4 to 11111010, and (3) processor A writes the updated data to the main memory.

(6 marks)

(d) From the C++ code excerpt shown in **Figure Q2(d)**, determine the output of variable X and Y at the end of the execution using sequential consistency model, assuming initial condition is X = 1 and Y = 1:

```
    process 0
1   x=2;
2
3   if (y=1)
4           x=4;
5
6   printf("x = %d\n", x);
```

```
process 1
y=2;

if (x=1)
        y=4;

printf("y = %d\n", y);
```

FIGURE Q2(d)

(4 marks)

Q3 (a) Distinguish between parallel and concurrent programming.

(4 marks)

(b) Assume that there are 10 cores in a computer that can be used to execute an application in parallel and 95% of application code is parallelizable. Determine whether it is possible to achieve speedup of 7? Calculate the numbers of cores that are needed.

(4 marks)

(c) From the code shown in **Figure Q3**, select the suitable OpenMP pragmas to be used:

```
 1  int main (int argc, char ** argv)
 2  {
 3          int i, n;
 4          int h, x, sum;
 5          n = atoi(argv[1]);
 6          h = 2;
 7          sum = 0;
 8
 9          for (i=0; i<=n; i++) {
10                  x = h * (I + 5);
11                  sum += (1 + x * x);
12          }
13
14          return h * sum;
15
16          if (sum = 10) {
17                  result = sum;
18          }
19  }
```

FIGURE Q3

(i) to get the number of threads in the parallel region

(2 marks)

(ii) to set the number of threads in the parallel region

(2 marks)

(iii) to get the unique identity for each thread in the parallel region

(2 marks)

(iv) to parallelize the *for loop* with shared variable *h*

(2 marks)

(v) to apply barrier synchronization before executing

(2 marks)

(d) From the C++ code shown in **Figure Q3(d)**, evaluate the output of the program when the number of threads is 4:

```
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main (void)
5  {
6          int counter;
7
8          counter = 111;
9
10         #pragma omp parallel default()none shared(counter)
11         {
12
13                 #pragma omp atomic
14                 counter++;
15
16                 printf("counter = %d\n", counter);
17         }
18
19         return 0;
20 }
```

FIGURE Q3(d)

(8 marks)

Q4 (a) Explain the Flynn's Taxonomy for computer architecture.

(4 marks)

(b) Argue whether it is possible to achieve superlinear speed up or not.

(4 marks)

(c) Describe the reasons to implement synchronization in parallel programming.
(3 marks)

(d) Detect if a race condition happens in the following C++ code shown in **Figure Q4**. Explain your answer.

```
1  for (i=0; i<100; i++)
2     for (j=0; j<100; j++)
3        a[i][j] = b[i][j] + c[i][j];
```

FIGURE Q4

(4 marks)

Q5 (a) Differentiate between Central Processing Unit (CPU) and Graphic Processing Unit (GPU) architecture using appropriate diagrams.
(5 marks)

(b) Explain TWO (2) limitations of GPU architecture.
(4 marks)

(c) Based on the Compute Unified Device Architecture (CUDA) code shown in **Figure Q5**, write comments for the function of each block.

```
1  int main ()
2  {
3        size_t size = n * sizeof(float);
4        float* d_a;
5        cudaMalloc((void**) &d_a, size);
6        float* d_b;                                          Block 1
7        cudaMalloc((void**) &d_b, size);
8        float* d_c;
9        cudaMalloc((void**) &d_c, size);
10
11       cudaMemcpy(d_a, h_a, size, cudaMemcpyHostToDevice);
12       cudaMemcpy(d_b, h_b, size, cudaMemcpyHostToDevice);  Block 2
13
14       int threadsPerBlock = 256;
15       int threadsPerGrid = (n + threadsPerBlock - 1) / threadsPerBlock;   Block 3
16       vecAdd<<<threadsPerGrid, threadsPerBlock>>>(d_a, d_b, d_c);
17
18       cudaMemcpy(h_c, d_c, size, cudaMemcpyDeviceToHost);  Block 4
19
20       cudaFree(d_a);
21       cudaFree(d_b);
22       cudaFree(d_c);
23 }
```

FIGURE Q5

(8 marks)

– END OF QUESTIONS –