Universiti Tun Hussein Onn Malaysia

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FINAL EXAMINATION
SEMESTER I
SESSION 2016/2017

COURSE NAME	:	COMPUTER SYSTEMS ENGINEERING
COURSE CODE	:	BEC41603
PROGRAMME	:	BEJ
EXAMINATION DATE	:	DECEMBER 2016 / JANUARY 2017
DURATION	:	2 HOURS AND 30 MINUTES
INSTRUCTION	:	ANSWER **ALL** QUESTIONS

THIS QUESTION PAPER CONSISTS OF **EIGHT (8)** PAGES

Q1 (a) Differentiate between pipeline and multicore processor architecture using appropriate diagrams.

(4 marks)

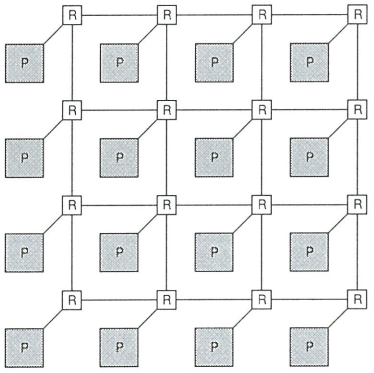
(b) Briefly explain the issue of bus architecture for multicore processor architecture in terms of long communication delay due to arbitration.

(4 marks)

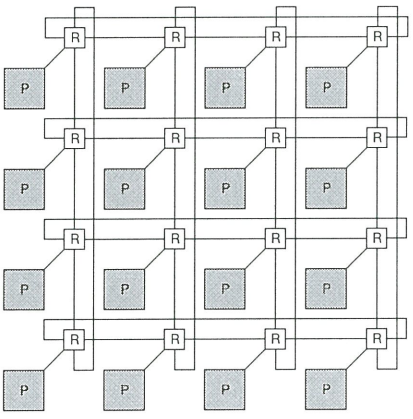
(c) List four standards of bus architecture and their corresponding developers.

(4 marks)

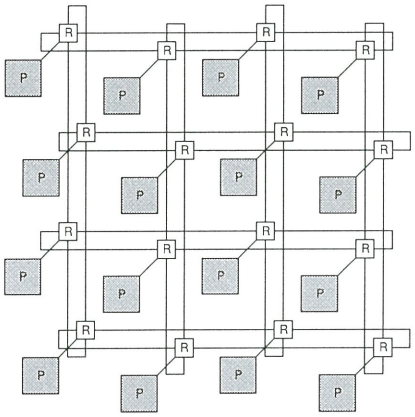
(d) Determine the network diameter for the Network on Chip (NoC) topology (i), (ii), (iii), and (iv) that are shown in **Figure Q1(d)**.



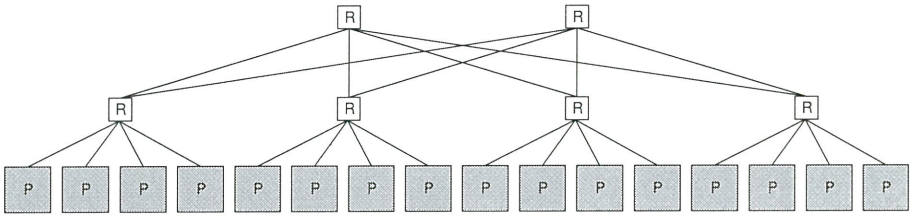
(i)



(ii)



(iii)



(iv)

R = router, P = processor.

Figure Q1(d)

(8 marks)

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

(4 marks)

(b) Explain the impact of cache coherency to the performance of multicore architecture as the number of processing cores increases.

(4 marks)

(c) Predict the states of memory location *0×0fd4* in cache A, *0×0430* in cache B, and *0×03dc* in cache C from **Figure Q2(c)** based on the following conditions using MESI coherency protocol.

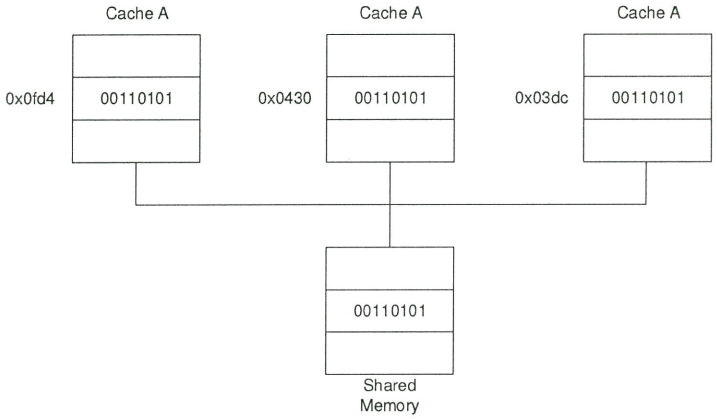


Figure Q2(c)

Conditions:

(1) Initially, *0×0fd4 = 00110101*, *0×0430 = 00110101*, *0×03dc = 00110101*
(2) Then, processor A modifies cache A address *0×0fd4* to *11111010*
(3) Next, processor A writes the updated data to the main memory.

(6 marks)

(d) Given the code snippet in **Figure Q2(d)**, determine output of *m* and *n* at the end of the execution based on the sequential consistency model. Justify your answer.

```
1   Int m = 0, n=0;
2
3   // Process 1                  // Process 2
4
5   m = 3;                        n = 3;
6
7   If (n=0) then m = 5;          If (m=0) then n = 4;
8   Print m;                      Print n;
```

Figure Q2(d)

(5 marks)

Q3 (a) Distinguish between parallel and concurrent programming.

(4 marks)

(b) Assume that a computer has 10 cores that can be used to execute an application in parallel and 95% of application code is parallelizable.

(i) Calculate the numbers of cores that are needed to achieve a speedup of 7.

(3 marks)

(ii) Determine whether it is possible to achieve a speedup of 7?

(1 mark)

(c) From the code shown in **Figure Q3(c)**, 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(c)

(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 superlinear speed up is possible.

(4 marks)

(c) Distinguish between parallel execution mechanism in superscalar processors and multicore processors.

(4 marks)

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

(4 marks)

(e) List two types of Mutual Exclusive (MUTEX) synchronization.

(2 marks)

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

(5 marks)

(b) Explain two limitations of GPU architecture.

(4 marks)

(c) Relate parallel processing methods in GPU with the Flynn's Taxonomy.

(3 marks)

(d) Write the correct CUDA programming codes for (1) – (3) in line 20, 23 and 26 in the code shown in **Figure Q5(d)**.

```
1  __global__ void multiply(int *a, int *b, int *c)
2  {
3                  *c = *a * *b;
4  }
5
6  int main(void)
7  {
8                  int a, b, c; // host copies of a, b, c
9                  int *d_a, *d_b, *d_c; // device copies of a, b, c
10                 int size = sizeof(int);
11
12                 // Allocate space for device copies of a, b, c
13                 cudaMalloc((void **)&d_a, size);
14                 cudaMalloc((void **)&d_b, size);
15                 cudaMalloc((void **)&d_c, size);
16
17                 a = 2; b = 7; // Setup input values
18
19                 // Copy inputs to device
20                 (1)
21
22                 // Launch kernel on GPU
23                 (2)
24
25                 // Copy result back to host
26                 (3)
27
28                 // Cleanup
29                 cudaFree(d_a); cudaFree(d_b); cudaFree(d_c);
30
31                 return 0;
32 }
```

Figure Q5(d)

(6 marks)

– END OF QUESTIONS –