Indian Institute of Information Technology Sri City, Chittoor

Term 3 Examination– June 2023

Computer Architecture

Maximum Marks: 30 | Time Duration: 90 mins
Date: 09th June 2023 | Course Code: CS0200

Instructions

1. **All questions are compulsory. All sub-parts of a question should be written together.**
2. **Attach the question paper with the answer sheet.**

Multiple Objective Questions

1. Which of the following miss occurs when the cache is empty?
   - **a. Cold**
   - b. Conflict
   - c. Capacity
   - d. All of the above

2. A 4-way set associative cache consists of a total of 64 blocks. The main memory contains 4096 blocks each consisting of 128 words. What would be the size of tag, set index, and block offset?
   - a. 1, 4, 7
   - b. 6, 6, 7
   - c. **8, 4, 7**
   - d. 7, 5, 7

3. What is the amount of time required to execute 50 instructions on a 5 stages pipeline where each stage requires 2 ms. Also determine the overall speed up.
   - a. 500 ms, 5
   - b. 500 ms, 4.6
   - **c. 108 ms, 4.6**
   - d. 108 ms, 5

4. Consider a pipeline having 4 phases with duration 60, 50, 90 and 80 ns. Calculate the pipeline cycle time.
   - a. 40
   - b. 100
   - c. 50
   - **d. 90**

5. The processor takes 10 clock cycles to complete a program. The corresponding pipeline processor uses 5 stages with execution time of 3, 5, 4,1,2 clock cycles respectively. What is the speed up when a very large number of instructions are executed?
   - a. 1
   - **b. 2**
   - c. 3
   - d. 4

6. In a symmetric multi-core systems all cores are:
   - a. Superscalar only with each having a different speed.
   - b. Multithreaded processors with each having a different register file.
   - c. Vector processors with each having a different ALU unit.
   - d. Identical and they can be of any processor type.

7. Invoking GCC with higher optimization levels like -02 or -03 can further improve program performance but on the risk of

a. Program size may increase and difficult to debug
b. Program size may decrease and difficult to debug
c. Program may indicate undesired behavior
d. None of the above

8. Which of the following indicates the difficulty of creating parallel processing programs.
   A. Scheduling
   B. Synchronization
   C. Load balancing
   D. Communication
   a. A and B
   b. B , C and D
   c. A, B, C
   d. All of the following
9. Match the following

I.	.data	P) Run-time heap.
II.	Compilers and assemblers	Q) relocatable object files
III.	Procedures	R) Initialized global and static C variables
IV.	brk	S) Strong symbols

   a. I - P, II- Q, III- S, IV - R
   b. I - R, II- Q, III- S, IV - P
   c. I - Q, II- P, III- S, IV - R
   d. I - Q, II- R, III- P, IV - S

10. Which of the following best suits for fine grained parallelism
    a. Low communication and synchronization overhead
    b. General-purpose parallel computers fall in this category
    c. Increases the communication and synchronization overhead
    d. Shared memory architectures are not suitable for fine-grained parallelism.

Descriptive Questions

1. a.

Consider a cache of 256 blocks of 16 words each. The main memory is addressable with a 16 bits address. Fill-in the following table and mention the size of tag, set index and block offset for Direct Mapping, 4-way Set Associative Mapping and 8-way Set Associative Mapping.

[3 Marks]

Address	**Tag**	**Set Index**	**Block Offset**
Direct Mapping	**4**	**8**	**4**
4-way Set Associative Mapping	**6**	**6**	**4**
8-way Set Associative Mapping	**7**	**5**	**4**

b.

What do you understand about the Hazards in pipelining? Explain data hazards by giving a suitable example. Also discuss a solution to handle this problem. [1+2+2 Marks]

c.

We have 2 designs D1 and D2; D1 has 5 pipeline stages with execution time of 3,2,4,5,3 ns, while D2 has 8 pipeline stages with each 2ns execution time. How much time can be saved using design D2 over design D1 for the execution of 100 instructions? [2 Marks]

Solution:

D1: [k+(n-1)]t = (5+99)*5 = 520ns

D2: [k+(n-1)]t = (8+99)*2 = 214 ns

Time saved = 520-214 = 306 ns

2.

a. Suppose we wish to write a procedure that computes the inner product of two vectors u and v. By doing the same sort of transformations we did in class to transform the abstract program combine1 into the more efficient combine4, we get the following code:

```
1   /* Inner product. Accumulate in temporary */
2   void inner4(vec_ptr u, vec_ptr v, data_t *dest)
3   {
4   long i;
5   long length = vec_length(u);
6   data_t *udata = get_vec_start(u);
7   data_t *vdata = get_vec_start(v);
8   data_t sum = (data_t) 0;
9
10  for (i = 0; i < length; i++) {
11  sum = sum + udata[i] * vdata[i];
12  }
13  *dest = sum;
14   }
```

Modify the code for **inner4** to unroll the loop by a factor k = 6. [3.5 Marks]

b. "Even the best compilers are thwarted by optimization blockers" can you list some optimization blockers and also suggest some remedies. [1.5 Marks]

3.

a. To build the executable, the linker must perform two major tasks. Briefly explain each task. [2 Marks]

b. Explain taxonomy of parallel processor architectures with the help of a diagram [2 Marks]

c. Briefly explain the idea of the optimizations named code motion and strength reduction where the programmer or compiler has to do regardless of processor

[1 Mark]

2.

a. **Solution** **7 new code lines are added at line no 6, 11, 13,14,15, 16, 17**

Each line instruction carries 0.5 marks so 7 X 0.5 = 3.5 Marks

```
1   /* Inner Product. 6 X 1 unrolling */
2   void inner_u6x1(vec_ptr u, vec_ptr v, data_t *dest)
3   {
4   long i;
5   long length = vec_length(u);
6   long limit = length-5;
7   data_t *udata = get_vec_start(u);
8   data_t *vdata = get_vec_start(v);
9   data_t sum = (data_t) 0;
10   /* Do 6 elements at a time */
11  for (i = 0; i < limit; i+=6) {
12   sum = sum + udata[i] * vdata[i]
13   + udata[i+1] * vdata[i+1]
14   + udata[i+2] * vdata[i+2]
15   + udata[i+3] * vdata[i+3]
16   + udata[i+4] * vdata[i+4]
17   + udata[i+5] * vdata[i+5];
18   }
19   /* Finish off any remaining elements */
20   for (; i < length; i++) {
21  sum = sum + udata[i] * vdata[i];
22  }
23  *dest = sum;
24  }
```

b. **Solution (Listing the optimization blockers 1 mark, 0.5 mark for remedy)**

Few optimization blockers are

a. procedure calls
b. Memory aliasing
c. Unnecessary memory references

Remedies : use of inline functions

3.

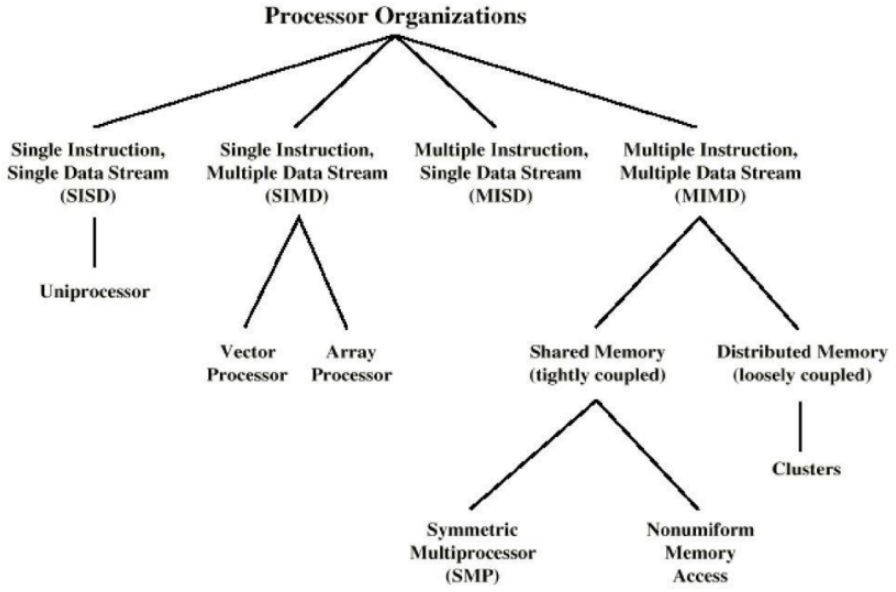
c. **Solution [listing the tasks 1 mark, brief description of each task 1 mark]**

Two major tasks linker performs to build the executable are

a. Symbol resolution
   - Programs define and reference symbols (global variables and functions
   - During the symbol resolution step, the linker associates each symbol reference with exactly one symbol definition.
b. Relocation

- Merges separate code and data sections into single sections
- Relocates symbols from their relative locations in the .o files to their final absolute memory locations in the executable.
- Updates all references to these symbols to reflect their new positions.

d. Total 12 classification are there , if they answer any 10 correct give 2 marks i.e, 0.2 x 10 = 2 marks



e. 0.5 mark for code motion 0.5 mark for strength reduction

Code motion *Identify a computation that is performed multiple times (e.g., within a loop), but such that the result of the computation will not change. We can therefore move the computation to an earlier section of the code that does not get evaluated as often.*

Strength reduction *Replace costly operation with simpler one*

Shift, add instead of multiply or divide

*16*x --> x << 4*