One goal of multiple internal buses is to simplify what process?

|  |  |
| --- | --- |
|  | Inter-process communication |
|  | Bus Arbitration |

|  |  |
| --- | --- |
|  | Parallel Processing |
|  | Process Management |

Suppose that you are working with a CISC machine using a 1.8 GHz clock (i.e., the clock ticks 1.8 billion times per second). This particular computer uses MASM-like instructions with the following timings:

add reg, mem 5 clock cycles (i.e., the ADD micro-program has 5 instructions)

add reg, immed 2 clock cycles

loop label 6 clock cycles

Suppose that the following code fragment is used to sum elements of a numeric array. For this problem, assume that memory limitations are non-existent and that there is no limit to the size of the array.

mov bx, 0 ;initialize sum

mov ecx, MAX\_SIZE ;initialize loop counter

mov esi, OFFSET list ;initialize array pointer

more:

add bx, esi ;add current list element

add \textrm{8}, 2 ;move array pointer to next element

loop more ;auto-decrement ecx, jump to more if ecx ≠ 0

After initialization, how many array elements can be processed in 3.9 ms? Round your answer to the nearest integer. Note that 1 ms. = 0.001 second.



Suppose you have a RISC machine with a 2.8 GHz clock (i.e., the clock ticks 2.8 billion times per second). This particular computer uses an instruction cache, a data cache, an operand fetch unit, and an operand store unit. The instruction set includes simple instructions with the following timings:

set reg, immed 3 clock cycle

loop label 6 clock cycles

add reg, immed 1 clock cycle

add reg, reg 4 clock cycles

load reg, mem 2 clock cycles

Assume that the following code fragment is used to sum the element of a numeric array. If the initialization code has already executed (i.e. the SET instructions have already finished execution) how many array elements can be processed in 3.7 ms? Round your answer to the nearest integer. Recall that 1 ms = 0.001 seconds. Also assume that there are no physical memory limitations, implying that the array can be as large as desired.

set r1, 0 ;initialize sum

set r2, MAX\_SIZE ;initialize loop counter

set r3, @list ;initialize array pointer

more:

load r4, \textrm{9} ;fetch current list element

add r1, r4 ;add current list element

add r3, 4 ;move array pointer to next element

loop more ;auto-decrement r2, jump to more if r2 != 0



Software parallelism is currently much more developed than hardware parallelism.

|  |  |
| --- | --- |
|  | True |
|  | False |

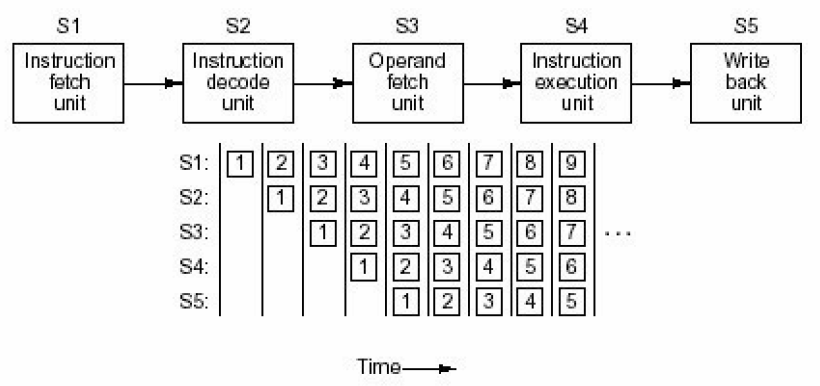
Assuming that all processor clock speeds are identical, executing a given software algorithm on a multicore processor is always faster than executing the same algorithm on a single-core processor.

|  |  |
| --- | --- |
|  | True |
|  | False |

Which of the following portions of a program can complicate the instruction-caching process? (Check all that apply)

|  |  |
| --- | --- |
|  | Repetition |
|  | Recursion |

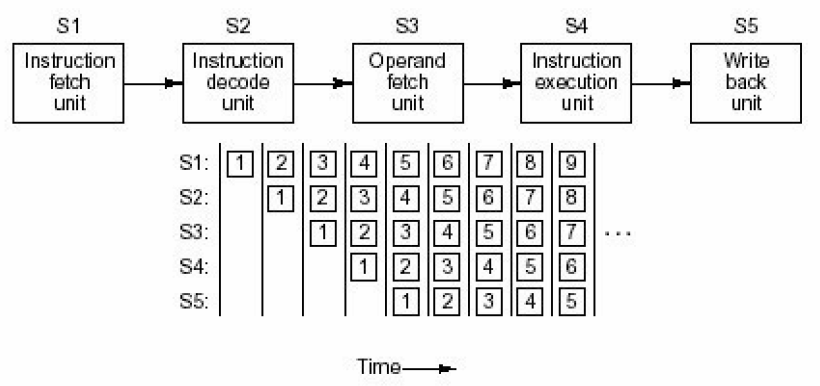
|  |  |
| --- | --- |
|  | Decision Structures |
|  | Sequential Execution |

Assume that you are working with the five-stage pipeline shown in the diagram.  
  
  
Suppose that each stage requires 3.7 nanoseconds to complete its task.  
How many nanoseconds would it take to complete 145 instructions *without* pipelining? Round your answer to the nearest integer.



An algorithm takes 6.2 seconds to execute on a single 2.6 GHz processor. 39% of the algorithm is sequential. Assume that there is zero latency and that the remaining code exhibits perfect parallelism.  
  
How long (in seconds) should the algorithm take to execute on a parallel machine made of 8 2.6 GHz processors? Round answers to one decimal place.



Assume that you are working with the five-stage pipeline shown in the diagram.  
  
Suppose that each stage requires 3.3 nanoseconds to complete its task.  
How many nanoseconds will it take to complete 106 instructions *with* pipelining? Round your answer to the nearest integer.



The CPU clock cycle length is the only contributing factor to the speed of operations on a computer.

|  |  |
| --- | --- |
|  |  |
|  | False |

Suppose that you are working with a CISC machine using a 2.2 GHz clock (i.e., the clock ticks 2.2 billion times per second). This particular computer uses MASM-like instructions with the following timings:

add reg, mem 6 clock cycles (i.e., the ADD micro-program has 6 instructions)

add reg, immed 3 clock cycles

loop label 4 clock cycles

Suppose that the following code fragment is used to sum elements of a numeric array. For this problem, assume that memory limitations are non-existent and that there is no limit to the size of the array.

mov bx, 0 ;initialize sum

mov ecx, MAX\_SIZE ;initialize loop counter

mov esi, OFFSET list ;initialize array pointer

more:

add bx, esi ;add current list element

add \textrm{8}, 2 ;move array pointer to next element

loop more ;auto-decrement ecx, jump to more if ecx ≠ 0

After initialization, how many array elements can be processed in 3.1 ms? Round your answer to the nearest integer. Note that 1 ms. = 0.001 second.



Suppose you have a RISC machine with a 1.7 GHz clock (i.e., the clock ticks 1.7 billion times per second). This particular computer uses an instruction cache, a data cache, an operand fetch unit, and an operand store unit. The instruction set includes simple instructions with the following timings:

set reg, immed 1 clock cycle

loop label 10 clock cycles

add reg, immed 3 clock cycle

add reg, reg 3 clock cycles

load reg, mem 4 clock cycles

Assume that the following code fragment is used to sum the element of a numeric array. If the initialization code has already executed (i.e. the SET instructions have already finished execution) how many array elements can be processed in 4.4 ms? Round your answer to the nearest integer. Recall that 1 ms = 0.001 seconds. Also assume that there are no physical memory limitations, implying that the array can be as large as desired.

set r1, 0 ;initialize sum

set r2, MAX\_SIZE ;initialize loop counter

set r3, @list ;initialize array pointer

more:

load r4, \textrm{7} ;fetch current list element

add r1, r4 ;add current list element

add r3, 4 ;move array pointer to next element

loop more ;auto-decrement r2, jump to more if r2 != 0

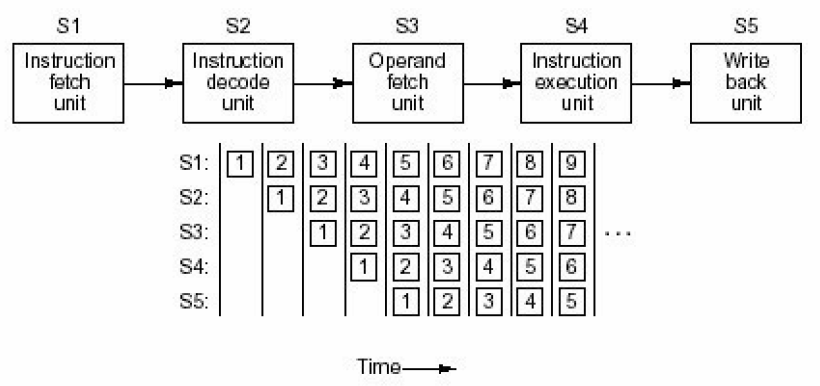


Multiprocessor Parallelism (shared memory) usually has lower coordination overhead than Multicomputer Parallelism (distributed memory).

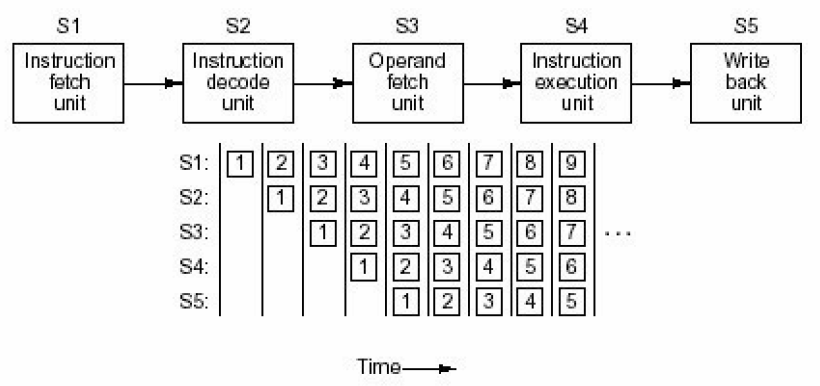
|  |  |
| --- | --- |
|  | True |

An algorithm takes 5.4 seconds to execute on a single 3 GHz processor. 48% of the algorithm is sequential. Assume that there is zero latency and that the remaining code exhibits perfect parallelism.  
  
How long (in seconds) should the algorithm take to execute on a parallel machine made of 6 3 GHz processors? Round answers to one decimal place.



Assume that you are working with the five-stage pipeline shown in the diagram.  
  
  
Suppose that each stage requires 3.8 nanoseconds to complete its task.  
How many nanoseconds would it take to complete 145 instructions *without* pipelining? Round your answer to the nearest integer.



Assume that you are working with the five-stage pipeline shown in the diagram.  
  
Suppose that each stage requires 3.7 nanoseconds to complete its task.  
How many nanoseconds will it take to complete 109 instructions *with* pipelining? Round your answer to the nearest integer.



Software parallelism is currently much more developed than hardware parallelism.

True

False