Started on	Thursday, 28 December 2023, 7:24 PM
State	Finished
Completed on	Thursday, 28 December 2023, 8:28 PM
Time taken	1 hour 4 mins
Marks	45.13/53.00
Grade	**51.09** out of 60.00 (**85.15**%)

Question **1**

Correct

Mark 0.50 out of 0.50

A processor should have, minimally, the following types of instructions, to fully utilize the functional units

- [x] a. Data Transfer ✔
- [x] b. Arithmetic and Logical ✔
- [x] c. Program Sequencing and Control ✔
- [x] d. I/O from I/O devices ✔
- [ ] e. Multi-media instructions
- [ ] f. Vector instructions
- [ ] g. Virtualization instructions

Question **2**

Correct

Mark 0.50 out of 0.50

Which of the following statements summarizes Amdahl's law most correctly?

- [x] a. The maximum speedup that can be achieved in a program, using parallel computing, is limited by the serial component of that program. ✔
- [ ] b. The maximum speedup that can be achieved in a program, using parallel computing, is limited by the number of processors.
- [ ] c. The maximum speedup that can be achieved in a program, using parallel computing, is limited by the parallel component of that program.
- [ ] d. The minimum speedup that can be achieved in a program, using parallel computing, is limited by the serial component of that program.
- [ ] e. The minimum speedup that can be achieved in a program, using parallel computing, is limited by the number of processors.
- [ ] f. The minimum speedup that can be achieved in a program, using parallel computing, is limited by the parallel component of that program.

Question **3**

Partially correct

Mark 0.50 out of 1.00

Using the notation and ISA for CISC, as mentioned in the textbook,

For each of these instructions, mention how many memory accesses (excluding the fetch, even for a multi-word instruction) they lead to:

Write a number in each box.

Add R5, R6: 0 ✔

And 10(R7), R9 : 1 ✖

Cmp 5(R7), 8(R9) : 2 ✔

Or 10(R7), 4(R9) : 1 ✖

Question **4**

Correct

Mark 2.00 out of 2.00

Given below are statements regarding MMU, Virtual Memory.

Mark them as True/False.

True	False		
◉☑	○✖	The physical address is the address of the actual physical memory location.	✔
○✖	◉☑	The MMU gets used for every instruction fetch, but not data.	✔
○✖	◉☑	When the kernel is executing, the conversion from logical to physical address is not required.	✔
◉☑	○✖	In a multi-tasking system, the MMU is setup by kernel, for each process, before the process gets scheduled.	✔
○✖	◉☑	Each logical address is converted to physical address by the kernel, using the MMU.	✔
◉☑	○✖	The instructions to set up MMU are privileged instructions.	✔
○✖	◉☑	Memory boundary violations are detected by the kernel.	✔
○✖	◉☑	The MMU gets used for only user processes, and not for kernel.	✔
◉☑	○✖	In a multi-tasking system, the MMU is set up by the kernel, for each process, because the kernel knows the actual physical memory layout of the process.	✔
◉☑	○✖	When a user process is executing, the convertion from logical to physical address is done by hardware.	✔
◉☑	○✖	The MMU gets used for every memory access.	✔
◉☑	○✖	The MMU converts the virtual address to a physical address.	✔
◉☑	○✖	Memory boundary violations are detected in hardware, by MMU.	✔
◉☑	○✖	The virtual address refers to the address as seen by the programmer/compiler and thus issued by CPU (from PC, or CS+EIP, etc)	✔

Question **5**

Correct

Mark 0.50 out of 0.50

In the following figure from the textbook

Block position	Contents of data cache after pass:								
	$j = 1$	$j = 3$	$j = 5$	$j = 7$	$j = 9$	$i = 6$	$i = 4$	$i = 2$	$i = 0$
0	A(0,0)	A(0,2)	A(0,4)	A(0,6)	A(0,8)	A(0,6)	A(0,4)	A(0,2)	A(0,0)
1									
2									
3									
4	A(0,1)	A(0,3)	A(0,5)	A(0,7)	A(0,9)	A(0,7)	A(0,5)	A(0,3)	A(0,1)
5									
6									
7									

Figure 8.21 Contents of a direct-mapped data cache.

The entry A(0,5) in j= 5 column means

- a. array entry A(0,5) existed in cache block 4, when j became = 5, during execution of the loop-of-j ✔
- b. array entry A(0,5) existed in cache block 4, before j became = 5, during execution of the loop-of-j
- c. array entry A(0,5) existed in cache block 4, throughout execution of the code
- d. array entry A(0,5) existed somewhere in cache, after j became = 5, in loop-of-j
- e. array entry A(0,5) existed somewhere in cache, before j became = 5, in loop-of-j

Question **6**

Correct

Mark 1.00 out of 1.00

Consider a multi-tasking operating system at work.Consider all the concepts we have studied *in this course*.

Assume, there are no compiler optimizaions.

Match the events related to memory used by a program, with the times at which they take place.

The address for the code of a function, relative to the beginning of the code is determined	Compile Time ✔
Value in a particular variable is determined	Run Time ✔
Physical memory location for the program's code is determined	Load Time ✔
Address for a local variable is determined, w.r.t. stack pointer	Compile Time ✔
Physical Memory location for the globals is determined	Load Time ✔
Whether the progrma has violated memory boundary is determined	Run Time ✔

Question **7**

Correct

Mark 1.00 out of 1.00

For each of the control signal, mention the number of select lines required

C_select: 2 ✔

MA_select: 1 ✔

RF_Write: Not applicable ✔

MFC: Not applicable ✔

Y_select: 2 ✔

B_select: 1 ✔

Question **8**

Partially correct

Mark 1.50 out of 2.00

Consider the following instructions at the given addresses in the memory:

```
1000 Mov R4, R2
1004 Add R4, R4, R4
1008 Add R2, R2, 1
1012 Add R5, R2, R4
```

Initially, registers R2 and R4 contain 20 and 30, respectively. These instructions are executed in a computer that has a five-stage pipeline as shown in Figure 6.5, and you can assume that both RZ and RY are data forwarded.

The first instruction is fetched in clock cycle 1, and the remaining instructions are fetched in successive cycles.

Note: for Mov instruction, the source can be treated as RA.
Write the contents of the registers, as specified below, after each mentioned cycle.

RA after cycle 4: 20 ✔

RA after cycle 5: 20 ✔

RY after cycle 5: 30 ✖

RY after cycle 6: 21 ✔

RZ after cycle 6: 41 ✖

R4 after cycle 6: 40 ✔

R2 after cycle 7: 21 ✔

R5 after cycle 8: 61 ✔

Question **9**

Correct

Mark 1.00 out of 1.00

Given below are some statements related to hardware threads, multi-core processors, superscalar processors.

Mark statements as True/False.

True	False		
◉ ☑	○ ✖	The SMT thread approach is most complex to implement.	✔
◉ ☑	○ ✖	The control unit is shared across multiple hardware-threads in one core	✔
◉ ☑	○ ✖	Superscalar processors helps in faster implementation of hardware multi-threading.	✔
◉ ☑	○ ✖	Hardware thread is characterized by CPU's capability to fetch multiple instruction sequences, and issue them	✔
◉ ☑	○ ✖	The SMT thread approach results in maximum throughput, among all approaches.	✔
○ ✖	◉ ☑	A cpu-core is equivalent to a hardware-thread.	✔
◉ ☑	○ ✖	Superscalar, without hardware multi-threading means the parallelism that can be exploited depends on ILP(Instruction Level Parallelism) inherent in the software instruction sequence	✔
○ ✖	◉ ☑	Hardware thread means entire pipeline will be replicated in CPU	✔
○ ✖	◉ ☑	VLIW captures TLP	✔
◉ ☑	○ ✖	Superscalar approach exploits ILP, while hardware thread approach exploits TLP.	✔
○ ✖	◉ ☑	Each hardware thread has it's own cache in CPU	✔
◉ ☑	○ ✖	It's possible to have hardware multi-threading even if CPU is not superscalar	✔

Question **10**

Partially correct

Mark 0.67 out of 2.00

```
            Move        R2, #NUM1    Push parameters onto stack.
            Subtract    SP, SP, #4
            Store       R2, (SP)
            Load        R2, N
            Subtract    SP, SP, #4
            Store       R2, (SP)
            Call        LISTADD      Call subroutine
                                       (top of stack is at level 2).
            Load        R2, 4(SP)    Get the result from the stack
            Store       R2, SUM        and save it in SUM.
            Add         SP, SP, #8   Restore top of stack
                                       (top of stack is at level 1).
            :

LISTADD:    Subtract    SP, SP, #16  Save registers
            Store       R2, 12(SP)
            Store       R3, 8(SP)
            Store       R4, 4(SP)
            Store       R5, (SP)       (top of stack is at level 3).
            Load        R2, 16(SP)   Initialize counter to n.
            Load        R4, 20(SP)   Initialize pointer to the list.
            Clear       R3           Initialize sum to 0.
LOOP:       Load        R5, (R4)     Get the next number.
            Add         R3, R3, R5   Add this number to sum.
            Add         R4, R4, #4   Increment the pointer by 4.
            Subtract    R2, R2, #1   Decrement the counter.
            Branch_if_[R2]>0 LOOP
            Store       R3, 20(SP)   Put result in the stack.
            Load        R5, (SP)     Restore registers.
            Load        R4, 4(SP)
            Load        R3, 8(SP)
            Load        R2, 12(SP)
            Add         SP, SP, #16    (top of stack is at level 2).
            Return                   Return to calling program.
```

Figure 2.18 Program of Figure 2.8 written as a subroutine; parameters passed on the stack.

W.r.t. the program given above

Calculate the contents of the stack pointer, SP, immediately after each of the following instructions in the program in Figure 2.18 (given above) is executed.

Assume that [SP] = 1000 at Level 1, before execution of the calling program begins.

Assume that link register is used for subroutine call-return.

(a) The First Store instruction in the subroutine. SP = 976 ✔

(b) The last Load instruction in the subroutine. SP = 980 ✖

(c) The last Store instruction in the calling program. SP = 996 ✖

Question **11**

Incorrect

Mark 0.00 out of 0.50

The basic idea behind designing a pipeline for execution of instructions is:

- a. To be able to achieve 1 instruction completion per cycle
- b. To simplify the design of the CPU hardware
- c. To teach students how a processor works
- d. To enable a smoother flow of instructions inside the processor ✖
- e. That's the only way to do it!
- f. To increase the cost of the processor
- g. To reduce power consumption

Question **12**

Correct

Mark 1.00 out of 1.00

Which of the following statements are correct, and which are incorerct, regarding Data Hazards and their handling?

Correct	Incorrect		
◉☑	○✖	values from RZ and RY registers are forwarded to both MuxA and MuxB to solve the problem	✔
◉☑	○✖	Software solutions for solving data hazards can sometimes lead to inefficiency, as the compiler may not be able to find instructions to fill the dependency gap.	✔
○✖	◉☑	Data Hazard can occur even if an earlier instruction has a source operand that is destination operand of a later instruction	✔
○✖	◉☑	In a 5-stage pipeline, data hazard can possibly occur with any two instructions that have 3 intermediate instructions between them.	✔
○✖	◉☑	In a 5-stage pipeline, data hazard can possibly occur with any two instructions that have 2 intermediate instructions between them.	✔
◉☑	○✖	Software solution can always be made to work, may be at the cost of efficiency.	✔

Question **13**

Correct

Mark 1.00 out of 1.00

Match cache enhancement mechanism with it's name.

A temporary space in cache, where writes gets recorded, before getting updated in RAM	Write Buffer ✔
Allow access to cache, while a miss is being served	Lockup free cache ✔
Used because writes can wait, but reads can not	Write Buffer ✔
Handle multiple outstanding cache misses at a time	Lockup free cache ✔
Instruction can be inserted by compiler, to tell processor to fetch next instruction in advance	Pre-fetch ✔

Question **14**

Correct

Mark 1.00 out of 1.00

Which of the following are the steps carried out by a kernel during boot process?

- [x] a. Setup it's own data structures ✔
- [x] b. Setup the IVT to point to appropriate kernel code locations for handling exceptions, h/w & s/w interrupts ✔
- [x] c. Create the first program, "init" (or systemd) by hand ✔
- [x] d. Setup MMU, timer for the execution of "init" program ✔
- [x] e. Pass control to the "init" program ✔
- [ ] f. First, Change the CPU mode to kernel mode
- [ ] g. Create all processes that are needed to run the GUI
- [ ] h. Load the compiler, linker into memory

Question **15**

Correct

Mark 1.00 out of 1.00

The number

123456789 stored as 32-bit integer on a Little Endian Machine, will be treated as the following number on a Big-Endian (32-bit) machine:

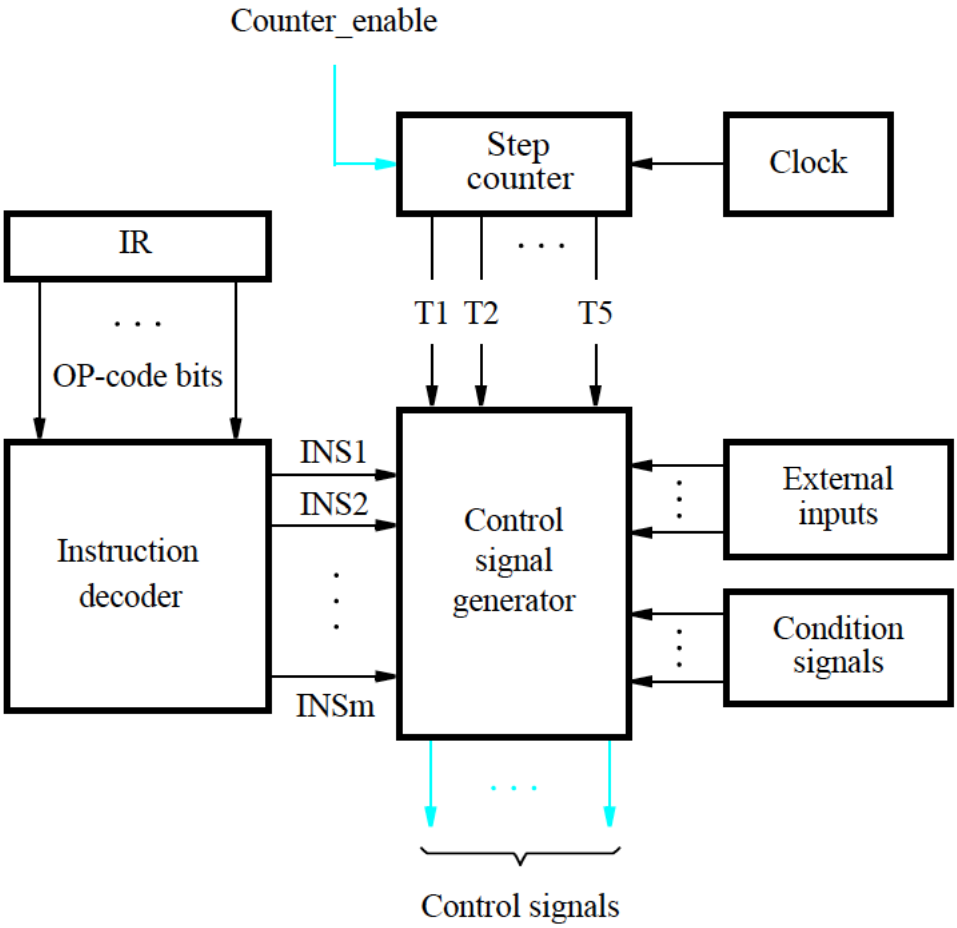
Answer: 365779719 ✔

Question **16**

Correct

Mark 1.00 out of 1.00

In this diagram



True	False		
○ ✗	● ✓	One or multiple of of INS_1 to INS_i lines is turned on by instruction decoder	✓
● ✓	○ ✗	External inputs includes Interrupts	✓
● ✓	○ ✗	External inputs include MFC signal	✓
● ✓	○ ✗	The Instruction decoder, is a combinational circuit	✓
● ✓	○ ✗	Only one of INS_1 to INS_i lines is turned on by instruction decoder	✓
○ ✗	● ✓	The Instruction decoder, is a sequential circuit	✓
○ ✗	● ✓	One of the lines, T1..T5 is turned on depending on the cycle, by the control signal generator	✓
● ✓	○ ✗	Condition Signals are provided by the Status register	✓
● ✓	○ ✗	One of the lines, T1..T5 is turned on depending on the cycle, by the step counter	✓

Question **17**

Correct

Mark 1.00 out of 1.00

Mark all points which bring out the difference between a micro-controller and a PC

- [x] a. Microcontrollers are often smaller size and consume less power, compared to PC ✔
- [x] b. Microcontroller processor chips are usually capable of running less numbers of instructions compared to PC ✔
- [x] c. Microcontrollers are usually employed for a dedicated specific purpose, and not general purpose ✔
- [x] d. It's possible that Microcontrolled processors do not have two modes of CPU operation (user/kernel mode), while multi-tasking PCs need processors with two modes. ✔
- [ ] e. A microcontroller is not a computer, while a PC is.
- [x] f. Examples of microcontrollers are microwave oven, digital camera, washing machine, etc. ✔
- [ ] g. Micro-controllers do not use a pipelined organization, while PCs use a pipelined organization

Question **18**

Correct

Mark 0.50 out of 0.50

Arrange the following in the correct boot order sequence

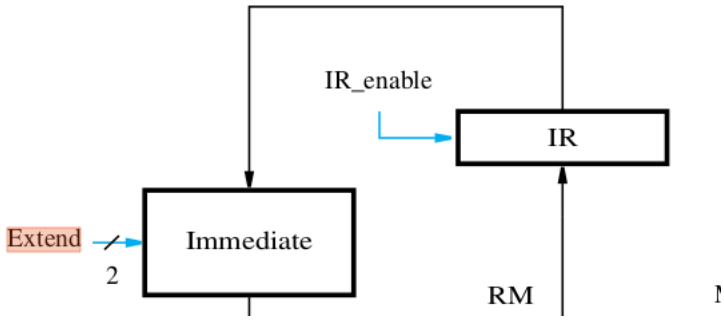
1. ✔ BIOS
2. ✔ Grub2
3. ✔ Linux Kernel
4. ✔ Init/systemd
5. ✔ Applications

Question **19**

Correct

Mark 1.00 out of 1.00

Consider the "Immediate" box from Figure 5.19



True	False		
○ ✗	◉ ✓	As per the textbook, the extend signal has 2 select lines because there are 4 possible ways of extending	✔
◉ ✓	○ ✗	The Extend control signal is to possibly extend the 'immediate' value extracted from IR	✔
○ ✗	◉ ✓	The output of extend box is fed to MuxINC, when the instruction is *not* a branch instruction	✔
◉ ✓	○ ✗	The output of extend box is fed to MuxINC, when PC-relative addressing is used	✔
◉ ✓	○ ✗	The output of extend box is fed to MuxB, when IR gives second operand of an instruction	✔
○ ✗	◉ ✓	The output of extend box is fed to MuxINC, when the instruction is *not* a branch instruction	✔
◉ ✓	○ ✗	This box exists, because of the particular way instruction encoding scheme has been designed in the textbook, allowing for sign extension of the offset	✔
◉ ✓	○ ✗	As per the textbook, the extend signal has 2 select lines because there are 3 possible ways of extending	✔

Question **20**

Correct

Mark 1.00 out of 1.00

Match LHS with RHS

Pipelined processor	= 1 instruction/cycle	✔
Hardware threaded processor	>1 instruction/cycle	✔
Non pipelined processor	< 1 instruction/cycle	✔
Superscalar processor	>1 instruction/cycle	✔

Question **21**

Correct

Mark 1.00 out of 1.00

Match device with speed (as in 2022!)

DDR4-3200W	25600 MB/S ✔
SSD SATA	450 MB/S ✔
DDR5	50 GB/S ✔
DDR4-1600	12800 MB/S ✔
NvME PCIE	6300 MB/S ✔
Typical SANDISK flash drive with USB3.0, available in market (e.g. SDIX70N-256G-GN6NE)	100 MB/S ✔
Typical 7200 RPM, SATA HDD	100 MB/S ✔

Question **22**

Incorrect

Mark 0.00 out of 3.00

Select all the correct statements about calling convention on x86 32-bit.

- [x] a. Parameters may be passed in registers or on stack (since this option is repeated, either select both or NONE) ✔
- [x] b. Paramters are pushed on the stack in left-right order ✖
- [ ] c. Compiler may allocate more memory on stack than needed
- [ ] d. Space for local variables is allocated by substracting the stack pointer inside the code of the called function
- [x] e. Space for local variables is allocated by substracting the stack pointer inside the code of the caller function ✖
- [x] f. The ebp pointers saved on the stack constitute a chain of activation records ✔
- [x] g. during execution of a function, ebp is pointing to the old ebp ✔
- [ ] h. Return address is one location above the ebp
- [ ] i. Parameters may be passed in registers or on stack (since this option is repeated, either select both or NONE)
- [x] j. The two lines in the beginning of each function, "push %ebp; mov %esp, %ebp", create space for local variables ✖
- [x] k. The return value is either stored on the stack or returned in the eax register ✖

Question **23**

Correct

Mark 1.00 out of 1.00

Match each functional block with it's purpose

IR	Store the currently executing instruction ✔
Instruction address generator	Update the PC ✔
Control Circuit	Generate signals to make other components perform desired actions ✔
Register file	Read/Write from/to registers ✔
ALU	Perform computation ✔

Question **24**

Correct

Mark 0.50 out of 0.50

Match the LHS with RHS

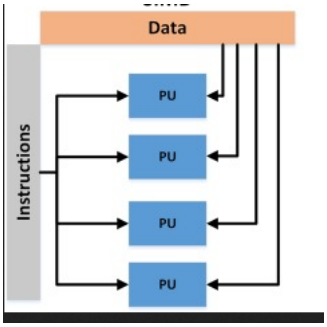
An operand is to be read from memory, and slower memory access makes instruction wait	Memory Hazard ✔
Instruction stalls because a source parameter is a destination of earlier instruction, which is not complete	Data Hazard ✔
Instruction is fetched, but not executed, because earlier branch instruction lead to another code path	Branch Hazard ✔

Question **25**

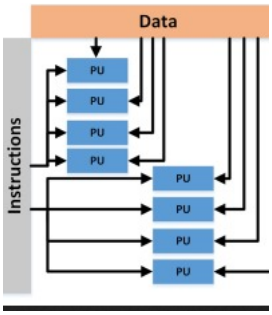
Correct

Mark 1.00 out of 1.00

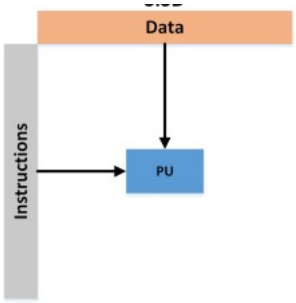
Match diagram with it's meaning



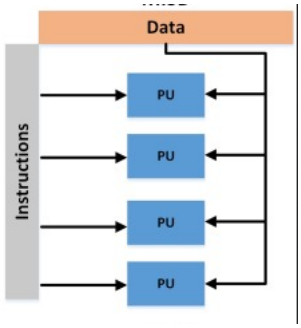
SIMD ✔



MIMD ✔



SISD ✔



MISD ✔

Question **26**

Correct

Mark 1.00 out of 1.00

Mark statements as True/False with reference to Execution Completion in a superscalar processor.

True	False		
○ ✗	◉ ☑	Reorder buffer is used by the commitment unit to reorder the instruction completion/commit sequence.	✔
○ ✗	◉ ☑	The register renaming technique is used for the commitment step, where another temporary copy of the registers from the register file is maintained, till the instruction is ready to commit.	✔
◉ ☑	○ ✗	The commitment step, the last step, is used to discard the results of an instruction (from being written to register file) if that instruction caused an exception.	✔
○ ✗	◉ ☑	The issue of Execution Completion arises because of the desire to have in-order completion of instructions.	✔
○ ✗	◉ ☑	Reorder buffer is used by the commitment unit to reorder the instruction execution sequence.	✔
◉ ☑	○ ✗	Reorder buffer is used by the commitment unit to sequentially retire instructions in the program order.	✔
◉ ☑	○ ✗	The issue of Execution Completion arises because of the desire to have precise exceptions.	✔

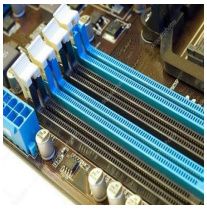
Question **27**

Correct

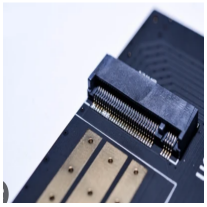
Mark 1.00 out of 1.00

Match pairs

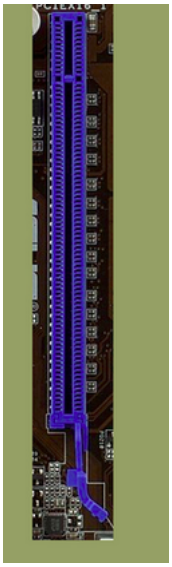
IDE Slot ✔



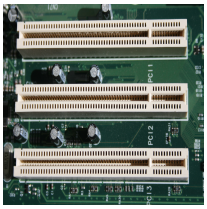
RAM Slot ✔



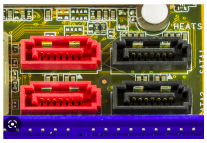
M2 slot ✔



PCIE 16 slot ✔



PCI slot ✔



SATA slot ✔

Question **28**

Correct

Mark 1.00 out of 1.00

Mark the statements related to Endian-ness as True/False.

True	False		
◉☑	○✖	Endian-ness is a property of the processor.	✔
○✖	◉☑	Endian-ness is a property of the RAM	✔
○✖	◉☑	Endian-ness is a property of the programming language.	✔
◉☑	○✖	You need not be worried about Endian-ness if all work you do ends up being done on the same computer.	✔
◉☑	○✖	Data to be sent on the network needs to be checked for Endian-ness because the computer at the end may have other endian-ness.	✔
◉☑	○✖	The compiler should be aware of the Endian-ness	✔
○✖	◉☑	Data to be sent on the network needs to be checked for Endian-ness because the physical network manipulates the byte-order.	✔
◉☑	○✖	If a binary file stored on a computer with an Intel processor is taken to a computer with a Motorola processor, then the file may not be read properly.	✔

Question **29**

Correct

Mark 1.00 out of 1.00

The AMD Ryzen 7 4700U (Family 23, Model 96)

Has Caches as follows:

L1 Instruction: 256 KiB ✔

L1 Data: 256 KiB ✔

L2: 4MB ✔

L3: 8MB ✔

and

Has 1 ✔ hardware threads per core.

Question **30**

Correct

Mark 1.00 out of 1.00

Before we discussed how Assembly/Machine code programs execute on CPU, in our discussion on Chapter-2 and Chapter-3, we made some assumptions. Which were those assumptions?

- a. The processor is a RISC processor✔
- b. All instructions have size of 32 bits✔
- c. Code and Data are already in RAM (the kernel, compiler, linker, loader, etc ensured this somehow)✔
- d. We do not care what happens after the last instruction of the program.✔
- e. The PC is pointing to our code✔

Question **31**

Correct

Mark 1.00 out of 1.00

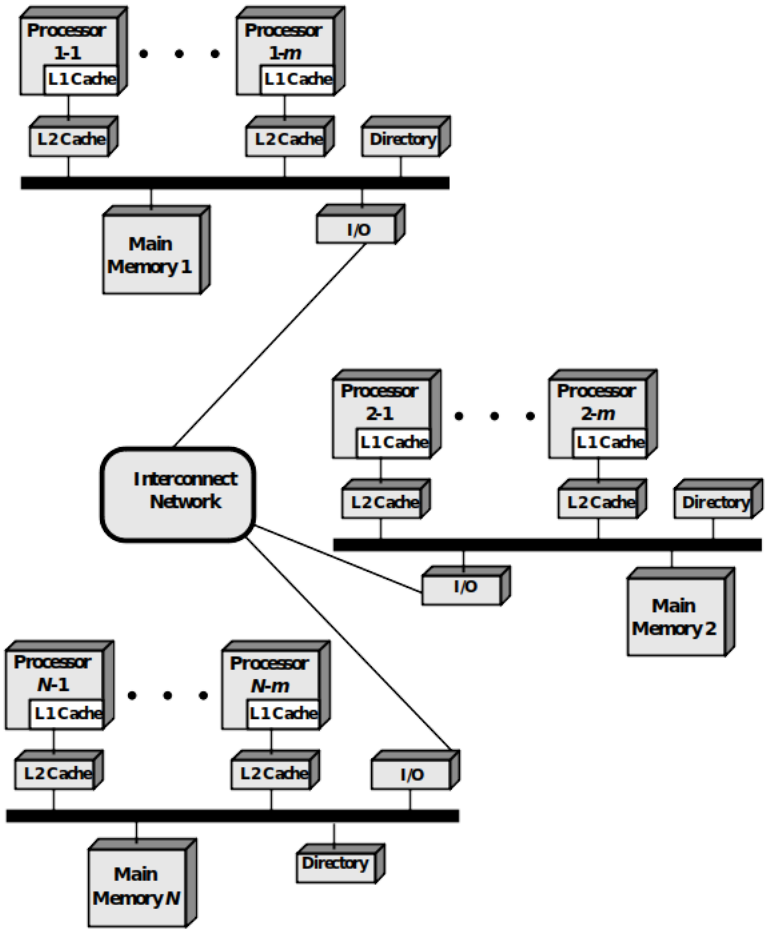


Figure 17.11 CC-NUMA Organization

With reference to above NUMA organization, the following example was discussed for accessing memory connected to a remote node in a NUMA system.

```
Suppose that processor 3 on node 2 (P2-3) requests a memory location 798, which is in the memory of node 1.  The following sequence
occurs:
1. P2-3 issues a read request on the snoopy bus of node 2 for location 798.
2. The directory on node 2 sees the request and recognizes that the location is in node 1.
3. Node 2’s directory sends a request to node 1, which is picked up by node 1’s directory.
4. Node 1’s directory, acting as a surrogate of P2-3, requests the contents of 798, as if it were a processor.
5. Node 1’s main memory responds by putting the requested data on the bus.
6. Node 1’s directory picks up the data from the bus.
7. The value is transferred back to node 2’s directory.
8. Node 2’s directory places the data back on node 2’s bus, acting as a surrogate for the memory that originally held it.
9. The value is picked up and placed in P2-3’s cache and delivered to P2-3.
```

Suppose, the above sequence of events has taken place. and now, once again P2-3 accesses the memory location 798.

How will this memory access take place?

a. The request will be satisfied from cache of P2-3✔

b. The request will be satisfied by the directory of node-2.

c. The request will be satisfied by the directory of node-1.

d. Exactly same sequence of actions will be carried out to satisfy the request

Question **32**

Partially correct

Mark 1.70 out of 2.00

Complete the details related to MESI protcol

	Modified	exclusive	Shared	Invalid
Memory copy is	Out of date ✔	Valid ✔	Valid ✔	Out of date ✖
Copies in other caches?	No ✔	No ✔	May be ✔	Yes ✖
Write will result in	Does not go to bus ✔	Goes directly to bus ✖	Goes to bus, updates cache ✔	Goes directly to bus ✔
State after Write Attempt on same node	Modified ✔	Modified ✔	Modified ✔	Modified ✔
State after Read Attempt on same node	Modified ✔	Exclusive ✔	Shared ✔	Shared ✔

Question **33**

Correct

Mark 1.00 out of 1.00

A block-set-associative cache consists of a total of 64 blocks, divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 32 words. Assuming a 32-bit byte-addressable address space, how many bits are there in each of the Tag, Set, and Word fields?

Word/Index: 5 ✔

Set: 4 ✔

Tag: 23 ✔

Question **34**

Correct

Mark 1.00 out of 1.00

Select all the correct statements about I/O handling.

- [x] a. In I/O mapped I/O, the CPU provides instructions like IN, OUT for communication with I/O devices. ✔
- [x] b. In Memory mapped I/O, certain parts of memory address space are mapped to I/O devices, so instructions like Load, Store can be used to communicate with I/O devices. ✔
- [x] c. Program controlled I/O is basically the busy wait mechanism, where the CPU(i.e. program) keeps running instructions checking if I/O device is ready. ✔
- [x] d. Interrupt controlled I/O means that the I/O device will inform the CPU that it wants to communicate, by raising an interrupt line. ✔
- [x] e. In Interrupt controlled I/O systems, when a program wants to do I/O, it must be "blocked" in the kernel, and some other program should get scheduled. ✔
- [ ] f. In multi-tasking systems, only Interrupt Controlled I/O code is part of the kernel, while Program Controlled I/O is part of user-applications.
- [ ] g. In Interrupt controlled I/O, the interrupt handler runs when the user process makes a request for I/O.
- [ ] h. IVT is the only way of handling multiple devices.
- [ ] i. Figure 3.8 is a good example of how interrupt handling works in a mult-tasking system.

Question **35**

Partially correct

Mark 0.88 out of 1.00

Complete the table, for the 4-state dynamic branch prediction algorithm.

Fill in the entries in the table for the "next-state" for a given current-state and event.

Current state->	SNT	LNT	ST	LT
Branch Taken	LNT ✔	ST ✔	ST ✔	ST ✔
Branch Not Taken	SNT ✔	SNT ✔	LT ✔	LNT ✖

Question **36**

Correct

Mark 1.00 out of 1.00

Compare the 3-bus interconnection network architecture, with the 5-stage pipeline architecture. Identify the statements which do the comparison correctly.

Correct	Incorrect		
✖	✔	Intel uses 3-bus architecture for it's CISC processors like i3.	✔
✔	✖	The bus architecture can use temporary registers in a more multi-purpose fashion, compared to pipeline architecture where the temporary registers have a fixed usage.	✔
✔	✖	3-bus interconnect suits CISC, while pipelined design suits RISC processors	✔
✔	✖	The 3-bus interconnect facilitates implementing instructions which need highly varying number of cycles, while the 5-stage pipeline facilitates implementing instructions which need nearly or exactly 5 stages to complete.	✔
✖	✔	It should always be easier to implement RISC processor using 3-bus architecture	✔
✔	✖	Each architecture has it's own stage-1, but stage-1 is the same for all instructions	✔
✖	✔	The clock is connected to each component in pipeline architecture, but not in 3-bus architecture.	✔

Question **37**

Partially correct

Mark 1.56 out of 2.00

Given below are some statements related to memory management in x86 in protected mode.

Marks the statements True/False

True	False		
○ ✖	◉ ☑	Address after segmentation is called logical address and address after paging is called linear address.	✔
◉ ✖	○ ☑	segmentation is done with the help of GDT and/or LDT table, both of which should be there in the processor	✖
○ ✖	◉ ☑	Page tables should be stored in memory, but page directory should be stored inside CPU	✔
◉ ☑	○ ✖	segmentation is mandatory, while paging is optional	✔
◉ ☑	○ ✖	All Page tables should exist in Memory	✔
◉ ☑	○ ✖	x86 supports both paging and segmentation.	✔
○ ☑	◉ ✖	segmentation is done with the help of GDT and/or LDT table, both of which should be there in Memory	✖
◉ ☑	○ ✖	The base address of page directory should be stored in CR3 register	✔
◉ ☑	○ ✖	Paging is supported with either 4MB pages or 4 KB pages.	✔

Question **38**

Partially correct

Mark 0.83 out of 1.00

With reference to this diagram

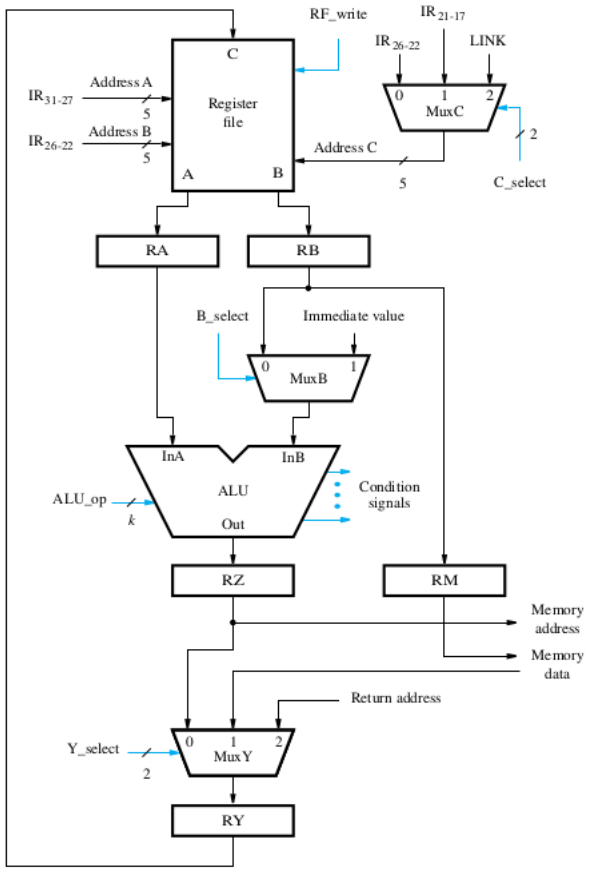


Figure 5.18 Control signals for the datapath.

Match each register with it's purpose

RZ	Result or ALU computation, or Data for Store instruction, or Return address after "Call" instruction	✖
RY	Result or ALU computation, or Data read in Load instruction, or Return address after "Call" instruction	✔
RB	Second source operand of an instruction	✔
RA	first source operand of an instruction	✔
InB	Second source operand of an instruction, or the immediate value from IR	✔
RM	Data for Store instruction	✔

Question **39**

Correct

Mark 1.00 out of 1.00

An SMP system is characterised by which of the following?

Yes	No		
◉☑	○✖	Different levels of cache may be local or shared among processors	✔
○✖	◉☑	A master processor communicates with all other processors to share access to cache	✔
◉☑	○✖	All processors run the same kernel (the kernel treats all processors in the same way)	✔
◉☑	○✖	Shared access to entire memory for all processors/cores	✔
◉☑	○✖	The "symmetric" refers to symmetricity in access to memory and execution of kernel code	✔
○✖	◉☑	I/O devices may or may not be shared equally among all processors	✔
◉☑	○✖	Uniform access to entire memory for all processors/cores	✔
◉☑	○✖	One or more processors/cores	✔
○✖	◉☑	Cache at all levels is shared among all the processors	✔
○✖	◉☑	Each processor is doing the same (symmetric) activity every point in time	✔

Question **40**

Correct

Mark 2.00 out of 2.00

We learnt one illustrative code demonstrating how x86 can transition from real mode to protected mode.

Mark the statements as True/False with respect to the code we discussed.

True	False		
○ ✗	◉ ☑	The GDT Table specifies the base as 1GB and Limit as 1GB for all the entries	✔
◉ ☑	○ ✗	Protection Enabled flag is set in the CR0 register	✔
◉ ☑	○ ✗	The (Address of GDT Table, Length of GDT Table) pair is loaded in GDTR	✔
◉ ☑	○ ✗	A GDT table exists in memory as part of the boot loader program	✔
◉ ☑	○ ✗	ljmp is called specifying new-value of CS, and new value of EIP as arguments.	✔
◉ ☑	○ ✗	The GDT table that exists in memory, has three entries.	✔
○ ✗	◉ ☑	The SEG_KCODE(=1) is shifted left by 3 bits, because the code segment should be set to 8.	✔
◉ ☑	○ ✗	the SEG_ASM Macro creates a 64 bit entry.	✔
○ ✗	◉ ☑	The GDT Table specifies a different base and different limit for each entry.	✔
◉ ☑	○ ✗	The SEG_KCODE(=1) is shifted left by 3 bits, because the least significant 3 bits of a segment selector are reserved (for TI and RPL)	✔

Question **41**

Partially correct

Mark 1.33 out of 2.00

Arrange the following events in handling a DMA request, in the correct order.

1. ✗ user code(process P1): does a function call like fread(fp,....) specifying large file read
2. ✗ Kernel code: (DMA driver code) Writes instructions in the DMA controller register, specifying #words, direction, and starting address.
3. ✓ kernel code: read() system call code runs. Identifies the disk from which the read should take place.
4. ✗ kernel code: (DMA driver code) prepares to issue DMA instructions, does appropriate calculations
5. ✓ kernel code: scheduler schedulers another process (say P2).
6. ✓ anything: Many other events can happen after this (except P1 running).
7. ✓ DMA interrupt occurs.
8. ✓ kernel code: DMA interrupt handler puts P1 on "ready-queue" of the scheduler.
9. ✓ kernel code: DMA interrupt handler returns (into whatever was happening before that)
10. ✓ some point in time Scheduler is invoked again (maybe as timer interrupt handler or some other way)
11. ✓ kernel code: Scheduler selects P1 for execution.
12. ✗ kernel code: P1 resumes read system call, returns in fopen() of P1
13. ✓ Kernel code: puts user process P1 on wait-queue.
14. ✗ kernel code: calls scheduler()
15. ✓ user code: P1 resumes after fopen()

Question **42**

Correct

Mark 1.00 out of 1.00

Write the booth multiplier for the bit-sequence given below.

1 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0

Multiplier is:

0 ✓ -1 ✓ 0 ✓ 0 ✓ +1 ✓ -1 ✓ +1 ✓ 0 ✓ -1 ✓

+1 ✓ 0 ✓ 0 ✓ 0 ✓ -1 ✓ 0 ✓ 0 ✓

Question **43**

Correct

Mark 1.00 out of 1.00

Match pairs.
Right-hand side numbers are in "bytes".

billion	1000000000 ✔
22MB	23068672 ✔
7Mb	917504 ✔
million	1000000 ✔
GB	1073741824 ✔
900KB	921600 ✔
KB	1024 ✔
MB	1048576 ✔

Question **44**

Correct

Mark 1.00 out of 1.00

The forwarding path in Figure 6.5 allows the contents of register RZ to be used directly in an ALU operation. The result of that operation is stored in register RZ, replacing its previous contents. This problem involves tracing the contents of register RZ over multiple
cycles. Consider the two instructions

```
I1 : Add R3, R2, R1
I2 : LShiftL R3, R3, #1
```

While instruction I1 is being fetched in cycle 1, a previously fetched instruction is performing an ALU operation that gives a result of 17. Then, while instruction I1 is being decoded in
cycle 2, another previously fetched instruction is performing an ALU operation that gives a result of 198. Also during cycle 2, registers R1, R2, and R3 contain the values 30, 100, and 45, respectively.

Using this information,
Write the value of register RZ as specified.

Register RZ after cycle 1:	17 ✔
Register RZ after cycle 2:	198 ✔
Register RZ after cycle 3:	130 ✔
Register RZ after cycle 4:	260 ✔

Question **45**

Correct

Mark 1.00 out of 1.00

Match the x86 register with the segment used with it.

esp	ss ✔
eip	cs ✔
ebp	ss ✔
esi	ds ✔
edi	es ✔

Question **46**

Partially correct

Mark 0.67 out of 1.00

Suppose that a computer has a processor with two L1 caches, one for instructions and one for data, and an L2 cache.

Let τ be the access time for the two L1 caches.

The miss penalties are approximately 10τ for transferring a block from L2 to L1, and 80τ for transferring a block from the main memory to L2.

Assume that the hit rates are the same for instructions and data and

that the hit rates in the L1 and L2 caches are 0.97 and 0.85, respectively.

Then, the average access time as seen by the processor is: 1.99 ✖ τ

Now, suppose L2 cache is removed, and L1 cache size is increased so that the hit rate in L1 becomes 0.99,

Then, the average access time as seen by the processor is: 1.79 ✔ τ

This leads to the conclusion that Removing L2, at the expense of better L1 is not a good idea ✔

Question **47**

Correct

Mark 1.00 out of 1.00

Mark the statements as correct/incorrect, with respect to the problem of branch delay and its solution

Correct	Incorrect		
◉☑	○✖	The Adder in Instruction Address Generator, can not be used to calculate the branch target, because this adder is used in every cycle to calculate address of next instruction	✔
◉☑	○✖	Static branch prediction is not a good idea, because branches are not usually taken randomly.	✔
◉☑	○✖	Relying on compiler to find an instruction to fill the branch delay slot, may not always work.	✔
○✖	◉☑	Conditional branches always lead to more penalties compared to unconditional branches.	✔
◉☑	○✖	Making an additional adder available in Decode stage, to do early calculation of branch target addres, reduces branch penalty by 1.	✔

◄ (Assignment) Dual boot your laptop. SKIP if already done and you know partitioning.

Jump to...

Shell Program Exercise ►