Computer Organization & Design ARM - review

1. **2^{16}**	65,536
2. **2^{36}**	68,719,476,736
3. **2^{64}**	18,446,744,073,709,551,616
4. **2^{8}**	256
5. **2 components of a computer**	datapath; control
6. **2 examples of Wireless network**	wifi, bluetooth
7. **2 facts about embedded computer**	-Hidden as components of systems -Stringent power/performance/cost constraints
8. **2 facts about supercomputers**	-High-end scientific and engineering calculations -Highest capability but represent a small fraction of the overall computer market
9. **2 features of instruction level parallelism**	-Hardware executes multiple instructions at once -Hidden from the programmer
10. **2 properties of a high level language**	-Level of abstraction closer to problem domain -Provides for productivity and portability
11. **3 additional ARMv8 features**	-Flexible second operand -Additional addressing modes -Conditional instructions (e.g. CSET, CINC)
12. **3 examples of non-volatile secondary memory**	-Magnetic disk -Flash memory -Optical disk (CDROM, DVD)
13. **3 Facts about servers**	Server computer -Network based -High capacity, performance, reliability -Range from small servers to building sized
14. **3 functions of an OS**	-Handling input/output -Managing memory and storage -Scheduling tasks & sharing resources
15. **3 layers of software/hardware?**	Compiler, assembler, hardware
16. **3 things that make parallel programming hard**	-Programming for performance -Load balancing -Optimizing communication and synchronization

17.	**3 ways to improve cpu time**	-Reducing number of clock cycles -Increasing clock rate -Hardware designer must often trade off clock rate against cycle count
18.	**4 things that impact cpu performance**	Algorithm: affects IC, possibly CPI Programming language: affects IC, CPI Compiler: affects IC, CPI Instruction set architecture: affects IC, CPI, Tc
19.	**4 Types of general addrressing in LEGv8**	1. Immediate 2. Register 3. Base 4. PC-relative
20.	**5 examples of progressing technology**	1951 - Vacuum tube - 1 1965 - Transistor - 35 1975 - Integrated circuit (IC) - 900 1995 - Very large scale IC (VLSI) - 2,400,000 2013 - Ultra large scale IC - 250,000,000,000
21.	**6 Steps of Procedure Calling**	1.Place parameters in registers X0 to X7 2.Transfer control to procedure 3.Acquire storage for procedure 4.Perform procedure's operations 5.Place result in register for caller 6.Return to place of call (address in X30)
22.	**31 x 32-bit general purpose sub-registers are...**	W0 to W30
23.	**31 x 64-bit general purpose registers are...**	X0 to X30
24.	**32-bit data called a...**	"word"
25.	**64-bit data is called a...**	"doubleword"
26.	**80386 is now known as...**	IA-32
27.	**A**	1010
28.	**Accumulator**	The accumulator is used to hold the result of operations performed by the arithmetic and logic unit, as covered in the section of the ALU.
29.	**Adding +ve and -ve operands will prevent...**	overflow

30.	**Addition and Subtraction**	These two tasks are performed by constructs of logic gates, such as half adders and full adders. While they may be termed 'adders', they can also perform subtraction via use of inverters and 'two's complement' arithmetic.
31.	**Address Bus**	The address bus contains the connections between the microprocessor and memory that carry the signals relating to the addresses which the CPU is processing at that time, such as the locations that the CPU is reading from or writing to. The width of the address bus corresponds to the maximum addressing capacity of the bus, or the largest address within memory that the bus can work with. The addresses are transferred in binary format, with each line of the address bus carrying a single binary digit. Therefore the maximum address capacity is equal to two to the power of the number of lines present (2^lines).
32.	**Addressing modes are...**	different ways to get data into registers (or store from registers into RAM)
33.	**Advantages of Harvard**	Since it had two memory locations, the allows parallel access to data and instructions. Data and instructions are accessed in the same way.
34.	**Advantages of RISC**	Speed. Since a simplified instruction set allows for a pipelined, superscalar design RISC processors often achieve 2 to 4 times the performance of CISC processors using comparable semiconductor technology and the same clock rates. Simpler hardware. Because the instruction set of a RISC processor is so simple, it uses up much less chip space. Smaller chips allow a semiconductor manufacturer to place more parts on a single silicon wafer, which can lower the per-chip cost dramatically. Shorter design cycle. Since RISC processors are simpler than corresponding CISC processors, they can be designed more quickly, and can take advantage of other technological developments sooner than corresponding CISC designs, leading to greater leaps in performance between generations. Efficient Code. Higher-level language compilers produce more efficient code than formerly because they have always tended to use the smaller set of instructions to be found in a RISC computer. Simplicity. The simplicity of RISC allows more freedom to choose how to use the space on a microprocessor.
35.	**Advantages of Von Nuemann**	Control unit gets data and instructions in the same way from memory. It simplifies design and development of the control unit. Data from memory and from devices are accessed in the same way. Memory organisation is in the hands of programmers.
36.	**Algorithm that is parallelizable**	Linear search
37.	**ALU**	Arithmetic and Logic Unit - Deals with all arithmetic and logic within the computer. The part of the central processing unit that deals with operations such as addition, subtraction, and multiplication of integers and Boolean operations. It receives control signals from the control unit telling it to carry out these operations.
38.	**Amdahl's Law**	(Execution time affected)/(Amount of improvement) + Execution time unaffected = Total Time
39.	**AND Operations are useful to...**	mask bits in a word
40.	**application software is written in...**	high-level language

41. **Are instruction count and CPI good performance indicators in isolation?**	No
42. **Arithmetic instructions use __ operands**	register
43. **ARM instructions are similar to that of...**	MIPS
44. **Array version of clearing an array requires shift to be...**	inside loop
45. **Assembler (or compiler) translates program into...**	machine instructions
46. **B**	1011
47. **A basic block is...**	No embedded branches (except at end), No branch targets (except at beginning)
48. **Bit 31 is __ bit**	sign
49. **Bit Shifting**	Shifting operations move bits left or right within a word, with different operations filling the gaps created in different ways. This is accomplished via the use of a shift register, which uses pulses from the clock within the control unit to trigger a chain reaction of movement across the bits that make up the word.
50. **Block virtualism**	This is the abstraction(separation) of logical storage from physical storage so that it may be accessed without the regard to physical storage or varied structure. This separation allows the administrators of the storage system greater flexibility in how they manage storage for end users.
51. **Both ARM and MIPS were announced in what year?**	1985
52. **C**	1100
53. **can faster division use parallel hardware like a multiplier?**	no
54. **Characteristics of CISC**	Complex instruction-decoding logic: It is driven by the need for a single instruction to support multiple addressing modes. Small number of general purpose registers: Instructions which operate directly on memory, and only the limited amount of chip space is dedicated for general purpose registers. Several special purpose registers: Many CISC designs set aside special registers for the stack pointer, interrupt handling, and so on. This can simplify the hardware design somewhat, at the expense of making the instruction set more complex. 'Condition code" register: This register reflects whether the result of the last operation is less than, equal to, or greater than zero and records if certain error conditions occur.

55.	**Characteristics of RISC**	Simple Instructions Limited fixed length instructions and no instructions combine load/store with arithmetic Few Data Types Supports simple data types such as integers/characterrs to complex data structures such as records Simple Addressing modes Use simple addressing modes and fixed length instructions to facilitate pipelining. Memory indirect addressing isn't provided. Identical general purpose Registers Allow any register to be used in any context Harvard Architecture Harvard memory model - The instruction stream and data stream are conceptually separated.
56.	**CISC**	Commonly implemented within large computers, this just uses one instruction to execute everything, instead of using multiple instructions.
57.	**Classifiying GPUs**	don't fit nicely into SIMD/MIMD model Static vs Dynamic and Instruction Level parallelism vs Data Level Parallelism
58.	**clock period is not always indicative of...**	performance
59.	**Clock Rate**	Frequency (X GHz) ($X \times 10^9$ Hz)
60.	**Coarse-grain multithreading**	Only switch on long stall Simplifies hardware, but doesn't hide short stalls
61.	**Comparison**	Comparison operations compare values in order to determine such things as whether one number is greater than, less than or equal to another. These operations can be performed by subtraction of one of the numbers from the other, and as such can be handled by the aforementioned logic gates. However, it is not strictly necessary for the result of the calculation to be stored in this instance.. the amount by which the values differ is not required. Instead, the appropriate status flags in the flag register are set and checked to determine the result of the operation.
62.	**Compiler optimizations are sensitive to...**	the algorithm
63.	**Compilers are good at making fast code from...**	simple instructions
64.	**Complex Instruction Set Architecture(CISC)**	The CISC approach attempts to minimize the number of instruction per program, sacrificing the number count per instruction.
65.	**conditional branches are potential for which type of hazard?**	control hazard
66.	**Control Bus**	The control bus carries the signals relating to the control and co-ordination of the various activities across the computer, which can be sent from the control unit within the CPU. Different architectures result in differing number of lines of wire within the control bus, as each line is used to perform a specific task. For instance, different, specific lines are used for each of read, write and reset requests.

67.	**Control Logic Circuits**	The control logic circuits are used to create the control signals themselves, which are then sent around the processor. These signals inform the arithmetic and logic unit and the register array what they actions and steps they should be performing, what data they should be using to perform said actions, and what should be done with the results.
68.	**Control Unit**	This controls the movement of instructions in and out of the processor, and also controls the operation of the ALU. It consists of a decoder, control logic circuits, and a clock to ensure everything happens at the correct time. It is also responsible for performing the instruction execution cycle.
69.	**CPI**	Clock Cycles Per Instruction
70.	**CPU**	Central Processing Unit - Brain of the computer; fetches, decodes and executes instructions.
71.	**CPU clock is used to sync...**	combinational logic
72.	**CPU Performance**	CPU Time = Seconds/Program = Instructions/Program X Cycles/Instructions X Seconds/Cycle. The CPU performance is dependent upon instruction Count, SPI (Cycles per Instruction) and Clock cycle time. All three are affected by the instruction set architecture.
73.	**D**	1101
74.	**Data Bus**	This is used for the exchange of data between the processor, memory and peripherals, and is bi-directional so that it allows data flow in both directions along the wires. Again, the number of wires used in the data bus (sometimes known as the 'width') can differ. Each wire is used for the transfer of signals corresponding to a single bit of binary data. As such, a greater width allows greater amounts of data to be transferred at the same time.
75.	**the datapath includes...**	registers
76.	**Decode**	Here, the control unit checks the instruction that is now stored within the instruction register. It determines which opcode and addressing mode have been used, and as such what actions need to be carried out in order to execute the instruction in question.
77.	**Decoder**	This is used to decode the instructions that make up a program when they are being processed, and to determine in what actions must be taken in order to process them. These decisions are normally taken by looking at the opcode of the instruction, together with the addressing mode used. This is covered in greater detail in the instruction execution section of this tutorial.
78.	**Defect rate determined by...**	manufacturing process
79.	**Design Principle 1:**	Simplicity favors regularity
80.	**Design Principle 2:**	Smaller is faster
81.	**Design Principle 3:**	Make the common case fast
82.	**Die area determined by...**	architecture and circuit design
83.	**Direct Addressing**	For direct addressing, the operands of the instruction contain the memory address where the data required for execution is stored. For the instruction to be processed the required data must be first fetched from that location.
84.	**direct mapped cache is...**	1 to 1
85.	**Disadvantages of Harvard**	Production of a computer with two buses and two memory storage's is more expensive and needs more time.

86.	**Disadvantages of RISC**	Code Quality. The performance of a RISC processor depends greatly on the code that it is executing. If the programmer (or compiler) does a poor job of instruction scheduling, the processor can spend quite a bit of time stalling: waiting for the result of one instruction before it can proceed with a subsequent instruction. Code Expansion. CISC machines perform complex actions with a single instruction; RISC machines may require multiple instructions for the same action, code expansion can be a problem. Code expansion refers to the increase in size that you get when you take a program that had been compiled for a CISC machine and re-compile it for a RISC machine. The exact expansion depends primarily on the quality of the compiler and the nature of the machine's instruction set. System Design. Another problem that faces RISC machines is that they require very fast memory systems to feed them instructions. RISC-based systems typically contain large memory caches, usually on the chip itself. This is known as a first-level cache.
87.	**Disadvantages of Von Neumann**	One bus has a bottleneck effect. Only one piece of information can be accessed at the same time. Instructions stored in the same memory as the data can be accidentally rewritten by and error in a program.
88.	**division operations ignore what two things**	overflow and division-by-zero
89.	**Dynamic data is...**	heap
90.	**dynamic linking automatically...**	picks up new library versions
91.	**dynamic linking avoids...**	image bloat caused by static linking of all (transitively) referenced libraries
92.	**dynamic linking requires...**	procedure code to be relocatable
93.	**E**	1110
94.	**Each part is ... of the other**	independent
95.	**EOR does the same operations that __ does**	ORR
96.	**EOR operations are also called...**	differencing operation
97.	**Execute**	The actual actions which occur during the execute cycle of an instruction depend on both the instruction itself, and the addressing mode specified to be used to access the data that may be required. However, four main groups of actions do exist, which are discussed in full later on.
98.	**Execution Cycle**	When a program is loaded into memory, it has to be executed.
99.	**Execution Time**	# of instructions × CPI × Clock Period OR #Clock Cycles×Clock period
100.	**F**	1111

101. **Fallacies**	Amdahl's law doesn't doesn't apply to parallel computers Peak performance tracks observed performance
102. **faster multiplication can be...**	pipelined
103. **Features of RISC**	One Cycle Execution Time: RISC processors have a CPI (clock per instruction) of one cycle. Pipelining: A technique that allows simultaneous execution of parts, or stages, of instructions to more efficiently process instructions. Large Number of Registers. The RISC design philosophy generally incorporates a larger number of registers to prevent large amounts of interactions with memory
104. **Fetch**	The fetch cycle takes the address required from memory, stores it in the instruction register, and moves the program counter on one so that it points to the next instruction.
105. **File Virtualism**	Addresses the NAS challenges by eliminating the dependencies between the data accessed at the file level and the location where the files are physically stored.
106. **Fine-grain multithreading**	switch threads after cycle interleave instruction execution if one thread stalls others are executed
107. **Flag register / status**	The flag register is specially designed to contain all the appropriate 1-bit status flags, which are changed as a result of operations involving the arithmetic and logic unit. Further information can be found in the section on the ALU.
108. **floating point standard was defined by...**	IEEE Std 754-1985
109. **floating point standard was last updated in...**	2008
110. **The following code is an example of what?** **LDR X2, [X0, #4]!**	immediate pre-indexed addressing
111. **the following code is an example of what?** **LDR X2, [X0],X1**	Register Post-Indexed Addressing Mode
112. **the following code is an example of what?** **LDR X2, [X0,X1, LSL#3]!**	Scaled Register Pre-Indexing
113. **For local data on the stack, there is a __ address and a __ address**	high, low
114. **for many years, we were getting ___ increases in CPU performance per year**	52%
115. **formula for cost per die**	(cost per wafer)/(dies per wafer * yield)
116. **formula for performance**	Performance = 1/Execution Time
117. **formula for yield**	1/(1 + (Defects per area * (die area/2))^2

118.	**formulas for cpu time**	cpu clock cycles x clock cycle time = cpu clock cycles/clock rate
119.	**formulas for die per wafer**	wafer area/die area
120.	**For nested call, caller needs to save what two things on the stack:**	Its return address Any arguments and temporaries needed after the call
121.	**For the occasional 32 bit constant, what 2 versions of mov do we use?**	MOVZ and MOVK
122.	**GHz numbers refer to...**	clock period
123.	**GPU**	GPUs are processors which can be used for a range of tasks other than processing computer game graphics. GPUs are used to display high quality video content such as HDMI or Blu-Ray on a screen. Video editing also requires many calculations, especially where edits or effects have been made. The decoding and encoding of videos is also carried out by the GPU
124.	**GPU**	-known as a co-processor. -traditionally responsible for the processing of large blocks of visual data very quickly.
125.	**GPU architecture**	-high number of cores, enabling it to handle multiple processes at once. -particularly good at processing multiple jobs in parallel.
126.	**GPU Architectures**	Processing is highly data parallel -GPUs are highly multithreaded -use thread switching to hide memory latency -Graphics memory is wide and high-bandwidth
127.	**Graphics and media processing operates on vectors of ___ and ___ data**	8-bit; 16-bit
128.	**Grid Computing**	Separate Computers interconnected by long-haul network
129.	**Harvard Architecture**	This is a computer architecture with physically separate storage and signal pathways for instructions of data.
130.	**Having more cores guarantees quicker processing time**	not always
131.	**the higher the yield, the ___ the chip**	cheaper
132.	**how are exceptions handled?**	save the PC of the offending instruction (using the ELR), and save the indication of the problem (using the ESR). instructions before the exception are saved, instructions after the exception are thrown away
133.	**how are locations determined in direct mapped cache?**	by the address
134.	**how can you increase the speed of a multiply?**	use multiple adders
135.	**how do conditional operations work in assembly?**	Branch to a labeled instruction if a condition is true. Otherwise, continue sequentially
136.	**how does a computer handle multiplication?**	via long multiplication - the length of the product is the sum of the operand lengths
137.	**how does an assembler help to translate a program into machine instructions?**	Provides information for building a complete program from the pieces

138. **how does integer subtraction actually function?**	Add negation of second operand
139. **how does linking object modules produce an executable image (3 steps)?**	1.Merges segments 2.Resolve labels (determine their addresses) 3.Patch location-dependent and external refs
140. **how does restoring division work**	Do the subtract, and if remainder goes < 0, add divisor back
141. **how does signed division work**	Divide using absolute values Adjust sign of quotient and remainder as required
142. **how does virtual memory work?**	uses a translation lookaside buffer to cache pages and speed up load times
143. **how do SISD and SIMD differ?**	SISD is where a processor executes a single instruction stream, to operate on data stored in a single memory; a SIMD processor performs a single, identical action simultaneously on multiple data pieces.
144. **how do the pieces of the datapath fit together?**	the memory stores the current instruction, the PC stores the address of the current instruction, the ALU executes the current instruction, and a mux chooses from multiple sources and steers one of those sources to its destination
145. **how do you resolve a control hazard?**	stalling or prediction
146. **how do you resolve a data hazard?**	forwarding, scheduling, or stalling (bubble)
147. **how is dynamic scheduling implemented?**	pipeline divided into 3 units: instruction fetch/issue unit, multiple functional units, and a commit unit. The first unit fetches instructions, decodes them, and and send to a functional unit. The functional units have reservation stations which holds the operands and operations. Once the buffer contains all its operands and the functional unit is ready to execute, the result is calculated. It is sent to any reservation stations waiting for this result, as well as the commit unit, which puts it in memory or a register.
148. **how is immediate addressing efficient?**	Efficient in regards to space and time, but only if the value fits in the 12-bit encoding scheme
149. **how is multiple issue implemented?**	static (decisions are made by the compiler before execution) or dynamic (decisions are made during execution by the processor)
150. **how is pipelining achieved?**	as soon as the resource is done with an instruction, it moves on to the next instruction, even if the first instruction hasn't gone through all the stages

151.	**how long can it take for a new chip to be manufactured?**	1-3 months
152.	**how many characters in ascii?**	128 (95 graphic, 33 control)
153.	**how many characters in latin-1?**	256 (ascii, plus 95 additional)
154.	**how many characters in unicode?**	32-bit character set (Used in Java, C++ wide characters)
155.	**how many data addressing modes in ARM?**	9
156.	**how many data addressing modes in MIPS?**	3
157.	**how many instructions can you do per cycle on a 12 stage pipeline?**	12
158.	**how many processing steps are there in manufacturing IC's**	20-40
159.	**how many registers in ARM?**	15 ×32-bit
160.	**how many registers in MIPS?**	31 ×32-bit
161.	**how many versions of immediate pre-indexed addressing are there?**	2 one where the address is added to the base and one where the address is subtracted from the base—to allow the programmer to go through the array forwards or backwards.
162.	**(IA-32) Hardware translates instructions to simpler...**	microoperations
163.	**IA-32 is a ___ instruction set**	complex
164.	**IA-32 is a microengine similar to...**	RISC
165.	**If a particular constant can not be represented by the defined 12 bit format, you get an...**	assembler error (invalid constant)
166.	**Immediate Addressing**	With immediate addressing, no look up of data is actually required. The data is located within the operands of the instruction itself, not in a separate memory location. This is the quickest of the addressing modes to execute, but the least flexible. As such it is the least used of the three in practice.
167.	**Immediate operand avoids...**	a load instruction
168.	**immediate pre-indexed addressing mode is useful for...**	going sequentially through an array
169.	**In an immediate offset, a constant address is...**	added to a base register. Example: LDRUSB X1, [X2,#1]
170.	**In an immediate pre-indexed load instruction, the content of the destination register changes _______ and the base register changes ______**	based on the value fetched from memory; to the address that was used to access memory.
171.	**in an optimized divider, there is __ cycle per partial-remainder subtraction**	One
172.	**in branch addressing, both addresses are...**	pc-relative

173. **in clearing an array, Array version requires shift to be...**	inside loop
174. **In clearing an array, the compiler can achieve same effect as...**	the manual use of pointers
175. **in determining how many n times faster one thing is to another, we use these formulas:**	performanceX/performanceY = exeuctionTimeY/executiontimeX = n
176. **Indirect Addressing**	When using indirect addressing, the operands give a location in memory similarly to direct addressing. However, rather than the data being at this location, there is instead another memory address given where the data actually is located. This is the most flexible of the modes, but also the slowest as two data look ups are required.
177. **induction variable elimination is better to...**	make the program clearer and safer
178. **in ic manufacturing, what is yield**	proportion of working dies per wafer
179. **in immediate addressing, the operand is...**	a constant within the instruction Example: ADD X2, X0, #5
180. **in immediate offset addressing, the constant offset is added to the _____ and then the result is used as the address in main memory to fetch from**	register inside the [ ]
181. **in immediate pre-indexed addressing mode, when do addition and subtraction occur?**	before the address is sent to memory.
182. **in integrated circuit production, relation to area and defect rate is...**	nonlinear
183. **in java, what interprets bytecode?**	the JVM
184. **in multiplication, there is ___ cycle per partial-product addition**	One
185. **in n-way set associative cache mapping, what is n, typically?**	the number of cores
186. **In optimized multiplication, what two steps are performed in parallel?**	add and shift
187. **In PC-relative addressing, the branch address is...**	the sum of the PC and a constant in the instruction. Example: B Loop1
188. **In PC-relative addressing the displacement from the PC is...**	signed (so branches can go forward or backward in the code)
189. **In reality most tasks are**	partially parallelizable
190. **in regards to immediate constants, large integers will....**	sometimes work
191. **In regards to immediate constants, the integers __ through ___ will always work.**	0, 4095
192. **In regards to prefix bytes, operation refers to what types of things?**	operand length, repetition, locking, etc.

193.	**In register addressing, the operand is...**	a register Example: ADD X2, X0, X1
194.	**in register offset addressing mod, another register is added to the ...**	base register.
195.	**In scaled register addressing, the register operand is...**	shifted first Example: ADD X2, X0, X1, LSL #3
196.	**Instruction Register**	This is used to hold the current instruction in the processor while it is being decoded and executed, in order for the speed of the whole execution process to be reduced. This is because the time needed to access the instruction register is much less than continual checking of the memory location itself.
197.	**Instructions are encoded in...**	binary
198.	**The Intel x86 ISA saw evolution in...**	backward compatibility
199.	**in virtual memory, which schema determines which block is going to be replaced?**	least-recently used (LRU)
200.	**I/O in ARM is...**	Memory mapped
201.	**Java/JIT compiled code is significantly faster than...**	JVM interpreted
202.	**LEGv8 has a __ x__ register file**	32 x 64-bit
203.	**LEGv8 instructions are Encoded as...**	32-bit instruction words
204.	**LEGv8 is typical of...**	RISC ISAs
205.	**LEGv8 register file is used for...**	frequency accessed data
206.	**linking object modules could leave location dependencies for fixing by...**	a relocating loader
207.	**List 3 things inside the processor**	-Datapath -Control -Cache memory

208.	**List the '8 Great Ideas'**	1.Design for Moore's Law 2.Use abstraction to simplify design 3.Make the common case fast 4.Performance via parallelism 5.Performance via pipelining 6.Performance via prediction 7.Hierarchy of memories 8.Dependability via redundancy
209.	**load/use hazard is a sub-type of which type of hazard?**	data hazard
210.	**Logic**	Problems that need to be solved, logically.
211.	**Logical Tests**	Further logic gates are used within the ALU to perform a number of different logical tests, including seeing if an operation produces a result of zero. Most of these logical tests are used to then change the values stored in the flag register, so that they may be checked later by separate operations or instructions. Others produce a result which is then stored, and used later in further processing.
212.	**Loosely Coupled Clusters**	Built of a network of independent computers -each has private memory and OS -Connected using high performance network system High availability, scalable, affordable, fault tolerant
213.	**Machine Language**	A mixture of electrical signals/string of binary bits that are unintelligible to humans.
214.	**main property of Volatile main memory**	Loses instructions and data when power off
215.	**Many ARMv8 instructions allow for 12 bit.....**	unsigned constants
216.	**Many compilers produce object modules....**	directly
217.	**Market share makes IA-32 economically...**	viable
218.	**Mean time to failure (MTTF)**	reliability measure. you want a HIGH MTTF
219.	**Memory**	The memory is not an actual part of the CPU itself, and is instead housed elsewhere on the motherboard. However, it is here that the program being executed is stored, and as such is a crucial part of the overall structure involved in program execution.

220. **Memory Address Register**	Used for storage of memory addresses, usually the addresses involved in the instructions held in the instruction register. The control unit then checks this register when needing to know which memory address to check or obtain data from.
221. **Memory Buffer/Data Register**	When an instruction or data is obtained from the memory or elsewhere, it is first placed in the memory buffer register. The next action to take is then determined and carried out, and the data is moved on to the desired location.
222. **Message Passing**	Each processor has private physical address space(clusters) Instructions/data sent to them Hardware sends/receives messages between processors
223. **MIMD**	Multiple Instruction Multiple Data Stream - Clusters
224. **MISD**	Multiple Instruction Single Data Stream- None
225. **Modeling Performance**	Assume performace metric of interest is achievable GFLOPs/sec Arithmetic Intensity of a kernel For a given computer, determine
226. **most caches use which kind of cache mapping?**	n-way set associative
227. **Most caches use which replacement policy?**	random
228. **Most constants are ___, and __-bit immediate is sufficient**	small; 12
229. **multicore microprocessors require...**	explicitly parallel programming
230. **Multicore processors**	-contains more than one processing unit -software must be written to specifically allow multiple jobs to be carried out simultaneously.
231. **Multicore Systems**	A multi-core processor is an integrated circuit to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks
232. **Multiple forms of addressing are generically called...**	addressing modes
233. **Multiplication and Division**	In most modern processors, the multiplication and division of integer values is handled by specific floating-point hardware within the CPU. Earlier processors used either additional chips known as maths co-processors, or used a completely different method to perform the task.
234. **Multithreading**	Performs multiple threads of execution in parallel
235. **Network Charateristics**	Performance -Latency per message -Throughput Cost Power Routability in Silicon
236. **Non-negative numbers have the same unsigned and ___ representation**	2s-complement

237.	**NUMA**	Non-Uniform Memory Access - is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to the processor. Under NUMA, a processor can access its own local memory faster than non-local memory (memory local to another processor or memory shared between processors). The benefits of NUMA are limited to particular workloads, notably on servers where the data are often associated strongly with certain tasks or users
238.	**One cycle per partial-product addition is ok if...**	frequency of multiplications is low
239.	**Opcode**	The opcode is a short code which indicates what operation is expected to be performed. Each operation has a unique opcode. Once the opcode is known, the execution cycle can occur. Different actions need to be carried out dependent on opcode, with two opcodes requiring the same actions to occur. 4 actions can occur: Transfer of data between CPU and memory.
240.	**Opcode Short Codes**	MOV Moves a data value from one location to another ADD Adds to data values using the ALU, and returns the result to the accumulator STO Stores the contents of the accumulator in the specified location END Marks the end of the program in memory
241.	**Operand**	The Operand indicates where the data required for the operation can be found and how it can be accessed.
242.	**Operating systems provide __ code**	service
243.	**Optimizing Perfomance**	Optimize fp performance (floating point?) -balance adds & multiplies Optimize Memory usage -Software prefetch -Memory Affinity
244.	**ORR operations are useful in...**	including bits in a word
245.	**Other general purpose registers**	These registers have no specific purpose, but are generally used for the quick storage of pieces of data that are required later in the program execution. In the model used here these are assigned the names A and B, with suffixes of L and U indicating the lower and upper sections of the register respectively.
246.	**Parallel**	The computational problem should be able to: Be broken apart into pieces of work that can be solved simultaneously; Execute multiple program instructions at any moment in time; Be solved in less time with multiple compute resources than with a single compute resource. The compute resources are typically: A single computer with multiple processors/cores An subjective number of such computers connected by a network

247.	**Parallel Computers**	Virtually all stand-alone computers today are parallel from a hardware perspective: (Multiple functional units (L1 cache, L2 cache, branch, fetch, decode, floating-point, graphics processing (GPU), integer, etc.) Multiple execution units/cores Multiple hardware threads
248.	**Parallel Computing**	This is the simultaneous use of multiple compute resources to solve a computational problem: A problem is broken into parts that can be solved concurrently Each part is further broken down to a series of instructions Instructions from each part execute simultaneously on different processors An overall control/coordination mechanism is employed
249.	**Parallelizable means....**	task can be broken unto separate process
250.	**Parallel processing**	processes are carried out at the same time.
251.	**Parallel Programming difficulties**	partioning, coordination, communications overhead(delay from message passing interface)
252.	**parallel programming for multicore microprocessors can compare with**	instruction level parallelism
253.	**Performance**	1÷(Execution time)
254.	**Performance Ratios**	(Performance A) ÷ (Performance B)
255.	**Pipelining**	In computers, a pipeline is the continuous and somewhat overlapped movement ofinstruction to the processor or in the arithmetic steps taken by the processor to perform an instruction. Pipelining is the use of a pipeline. Without a pipeline, a computer processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle. It must wait until it gets the next instruction. With pipelining, the computer architecture allows the next instructions to be fetched while the processor is performing arithmetic operations, holding them in a buffer close to the processor until each instruction operation can be performed. The staging of instruction fetching is continuous. The result is an increase in the number of instructions that can be performed during a given time period. Pipelining is sometimes compared to a manufacturing assembly line in which different parts of a product are being assembled at the same time although ultimately there may be some parts that have to be assembled before others are. Even if there is some sequential dependency, the overall process can take advantage of those operations that can proceed concurrently. Computer processor pipelining is sometimes divided into an instruction pipeline and an arithmetic pipeline. The instruction pipeline represents the stages in which an instruction is moved through the processor, including its being fetched, perhaps buffered, and then executed. The arithmetic pipeline represents the parts of an arithmetic operation that can be broken down and overlapped as they are performed. Pipelines and pipelining also apply to computer memory controllers and moving data through various memory staging places.
256.	**Pitfalls**	Not developing the software to take account of a multiprocessor architecture
257.	**Pointers correspond directly to...**	memory addresses

258.	**pointers help avoid...**	indexing complexity
259.	**Postfix bytes specify...**	addressing mode
260.	**Prefix bytes modify...**	operation
261.	**Pre-indexing constants must fit in .....**	9 bits (including the sign bit) (so -256 to 255)
262.	**Procedure call: ___ and ___**	branch and link
263.	**procedure calls are potential for which type of hazard?**	control hazard
264.	**Procedure return: jump register RET or BR LR What will this do?**	Copies LR to program counter Can also be used for computed jumps
265.	**procedure returns are potential for which type of hazard?**	control hazard
266.	**A Program can be loaded into absolute location in...**	virtual memory space
267.	**Program Counter**	This register is used to hold the memory address of the next instruction that has to executed in a program. This is to ensure the CPU knows at all times where it has reached, that is able to resume following an execution at the correct point, and that the program is executed correctly.
268.	**Progress in computer technology has been underpinned by...**	Moore's Law
269.	**purpose of X0 -X7**	procedure arguments/results
270.	**purpose of X8**	indirect result location register
271.	**purpose of X9 -X15**	temporaries
272.	**purpose of X16 -X17 (IP0 -IP1):**	may be used by linker as a scratch register, other times as temporary register
273.	**purpose of X18:**	platform register for platform independent code; otherwise a temporary register
274.	**purpose of X19 -X27:**	saved
275.	**purpose of X28 (SP):**	stack pointer
276.	**purpose of X29 (FP):**	frame pointer
277.	**purpose of X30 (LR):**	link register (return address)
278.	**purpose of XZR (register 31):**	the constant value 0
279.	**Quantum Computing**	Quantum computing studies theoretical computation systems (quantum computers) that make direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. Quantum computers are different from digital computers based on transistors.
280.	**Range of Signed ints**	-2,147,483,648 to 2,147,483,647

281. **Range of Unsigned ints**	0 to 4,294,967,295
282. **Reduced Instruction Set Architecture(RISC)**	RISC does the opposite, reducing the cycles per instruction at the cost of instruction per program.
283. **Register Array**	This is a small amount of internal memory that is used for the quick storage and retrieval of data and instructions. All processors include some common registers used for specific functions, namely the program counter, instruction register, accumulator, memory address register and stack pointer.
284. **Register offset addressing can help with indexing into an array, where the array index is...**	in one register and the base of the array is in another
285. **Register offset addressing mode can help with indexing into an...**	array
286. **Regularity makes implementation...**	simpler
287. **RISC**	is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures. Prime difference between RISC and CISC design is the number and complexity of instructions. CISC designs includes complex instruction sets so as to provide an instruction set that closely supports the operations and data structures used by Higher-Level Languages
288. **RISCV is almost the same architecture as...**	ARM
289. **saturating operations uses what sort of arithmetic?**	2s-complement modulo arithmetic
290. **scaled register offset addressing allows the register to be...**	shifted before it is added to the base register.
291. **Semantic GAP**	With an objective of improving efficiency of software development, several powerful programming languages have been developed. They provide high level of abstraction, conciseness and power. By this evolution the semantic gap grows. To enable efficient compilation of high level language programs, CISC and RISC designs are the two options. CISC designs involve very complex architectures including a large number of instructions and addressing modes, whereas RISC designs involve simplified instruction set and adapt it to the real requirements of user programs.
292. **Serial Computing**	A serial computer is typified by bit-serial architecture — i.e., internally operating on one bit or digit for each clock cycle. Machines with serial main storage devices such as acoustic or magnetostrictive delay lines and rotating magnetic devices were usually serial computers.
293. **Serial Computing**	A problem is broken into a discrete series of instruction. Instructions are executed sequentially one after another and executed on a single processor. Only one instruction may execute at any moment in time.
294. **Shared Memory**	SMP: shared memory multiprocessor -Hardware provides single physical address space for all processors -Synchronize shared variables using locks -Memory Access time -UMA vs NUMA

295. **SIMD**	Single Instruction Multiple Data Stream - GPU's, SSE instructions of x86 Operate element-wise on vectors of data All processors execute same instruction at the same time with different data addresses Simplifies synchronization Reduced Instruction Control Hardware Works best for highly data-parallel applications
296. **simplicty enables...**	higher performance at lower cost
297. **SISD**	Single Instruction Single Data Stream -Pentium 4
298. **SMT**	simultaneous multi-threading In multiple-issue dynamically scheduled processor -schedule instructions from multiple threads -instructions from independent threads execute when function units are available -within threads, dependencies handled by scheduling and register renaming
299. **The speed of processing depends on if program was written to .... of multiple cores**	take advantage
300. **SPMD**	Single Program multiple data, parallel program on a MMD computer -conditional code for different processors
301. **Stack is...**	automatic storage
302. **Static data contains...**	global variables
303. **Storage Virtualization**	Storage systems typically use special hardware and software along with disk drives in order to provide very fast reliable storage for computing and data.
304. **strided access**	accessing the same amount of bytes every time
305. **Subtracting +ve from -ve operand will overflow if...**	result sign is 0
306. **Subtracting two +ve or two -ve operands will result in...**	no overflow
307. **Subtracting -ve from +ve operand will overflow if...**	result sign is 1
308. **Syntax of loading a byte**	LDURSB Xt, [Xn, offset] (Sign extend to 64 bits in Xt (can be W or X))
309. **Syntax of loading a halfword**	LDURSH Xt, [Xn, offset] (Sign extends to 64 bits in Xt (can be W or X))
310. **Syntax of storing a byte**	STURB Wt, [Xn, offset] (Stores just rightmost byte of Wt (MUST be W and not X))
311. **syntax of storing a halfword**	STURH Wt, [Xn, offset] (Store just rightmost halfword of Wt (MUST be W and not X)
312. **System Bus**	This is comprised of the control bus, data bus and address bus. It is used for connections between the processor, memory and peripherals, and transferal of data between the various parts.
313. **system software is a...**	compiler
314. **Task that isn't parallelizable**	Fibonacci sequence

315.	**T/F - is the offset optional in immediate offset addressing?**	true (written as LDRUSB X1, [X2])
316.	**Timer or Clock**	The timer or clock ensures that all processes and instructions are carried out and completed at the right time. Pulses are sent to the other areas of the CPU at regular intervals (related to the processor clock speed), and actions only occur when a pulse is detected. This ensures that the actions themselves also occur at these same regular intervals, meaning that the operations of the CPU are synchronized.
317.	**time/units of work is also known as...**	raw speed/latency
318.	**Translate the following C Code into ARM: C code: f = (g + h) -(i + j); note: f, ..., j in X19, X20, ..., X23**	ADD X9, X20, X21 ADD X10, X22, X23 SUB X19, X9, X10

319.	**translate the following to assembly:
	
	if (a > b)
	a += 1;
	a in X22, b in X23**
		SUBS X9,X22,X23 // use subtract to make comparison
		
		B.LE Exit // conditional branch
		
		ADDI X22,X22,#1
		
		Exit:

320.	**UMA**	Uniform Memory Access -shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly. In a UMA architecture, access time to a memory location is independent of which processor makes the request or which memory chip contains the transferred data.
321.	**units of work/time is also known as...**	throughput/bandwidth
322.	**uses of GPU's**	due to the large number of cores many jobs are taken on by the GPU such as: -machine learning (AI) -modelling -Cryptocurrency mining (e.g. mining for bitcoins)
323.	**use ___ to improve performance**	parallelism
324.	**Vector Instructions**	lv,sv (load/store vector) addv.d add two vectors of double addvs.d add scalar to each element of double
325.	**Vector Processors**	Highly pipelined function units in CPU Stream data from/to vector registers to units elimates loops

326. **Vector vs. Multimedia Extensions**

Vector instructions have a variable vector length
multimedia extensions have a fixed width/length
vector instructions support strided access
vector units can be combination of pipelined and arrayed function units

327. **Vector vs. Scalar**

Vector Architectures and vectorizing compilers
-simplify data parallel programming
-explicit statement of absence of loop-carried dependencies
-regular access patterns benefit from interleaved and burst memory
-avoid control hazards by avoiding loops
More general than ad-hoc media extensions
-better match with compiler technology

328. **virtual memory uses which kind of cache mapping?**

fully associative

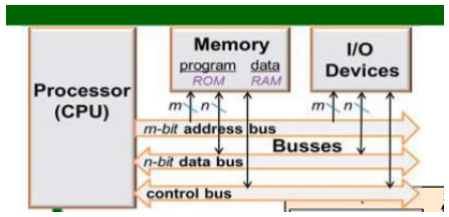
329. **virtual memory uses write-through or write-back?**

write-back

330. **Virtual Storage**

Virtual storage is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console.

331. **Von Neumann Architecture**



This describes the design architecture for an electronic digital computer with parts consisting of a processing unit containing:
ALU, Control Unit, Register Array, Memory to store both data and instructions, External Mass Storage, Input and Output.
Programs consist of a sequence of instructions. Instructions are executed in order they are stored in memory. Instructions, characters, data and numbers are represented in binary form.

332. **Von Neumann Continued**

All parts of the computer are connected together by Bus. Memory and devices are controlled by CPU. Data can pass through bus to and from CPU.
Memory holds both programs and data. Memory is addressed linearly; this means that there is an address for each and every memory location. Memory is addressed by the location number without regard to the data contained within.

333. **Von Neumann vs Harvard**

With Von Neumann architecture the CPU can be either reading an instruction or reading/writing data from/to the memory.
Both cannot occur at the same time since the instructions and data use the same bus system.
In a computer using the Harvard architecture, the CPU can both read an instruction and perform a data memory access at the same time, even without a cache.
A Harvard architecture computer can thus be faster for a given circuit complexity because instruction fetches and data access do not contend for a single memory pathway.
Also, a Harvard architecture machine has distinct code and data address spaces.

334. **Wafer cost and area are...**

fixed

335. **we make our technology "smaller and faster" with which of the 8 great ideas?**

abstraction - it decomposes ideas

336. **We use MOVZ and MOVK with with flexible....**

second operand (shift)

337.	**what 2 things will the following code do:** **BL ProcedureLabel**	Address of following instruction put in X30 Jumps to target address
338.	**What 3 things determine number of machine instructions executed per operation**	Programming language, compiler, architecture
339.	**what approach do computers take with division?**	long division
340.	**what are java class files?**	Simple portable instruction set for the JVM
341.	**what are non-leaf procedures**	Procedures that call other procedures
342.	**what are the 2 facets of performance?**	time/units of work and units of work/time
343.	**what are the 2 floating point representations**	Single precision (32-bit) Double precision (64-bit)
344.	**what are the 2 LEGv8 division operations?**	SDIV (signed) UDIV (unsigned)
345.	**what are the 2 types of locality and how do they differ?**	temporal locality: items accessed recently are likely to be accessed again soon. spatial locality: Items near those accessed recently are likely to be accessed soon
346.	**what are the 3 different ways of mapping cache?**	direct mapped cache, n-way set, and fully associative
347.	**what are the 3 LEGv8 multiple instructions and how do they function?**	MUL: multiply (Gives the lower 64 bits of the product) SMULH: signed multiply high (Gives the upper 64 bits of the product, assuming the operands are signed) UMULH: unsigned multiply high (Gives the upper 64 bits of the product, assuming the operands are unsigned)
348.	**what are the 3 main types of hazards?**	structure, data, and control
349.	**What are the 4 design principles?**	1.Simplicity favors regularity 2.Smaller is faster 3.Make the common case fast 4.Good design demands good compromises
350.	**What are the 5 integer operations?**	addition, subtraction, multiplication, division, handling overflow
351.	**what are the 5 steps for pipelining?**	fetch, decode, execute, read access, write back

352. **What are the 6 steps in the process of loading from an image file on disk into memory?**	1.Read header to determine segment sizes 2.Create virtual address space 3.Copy text and initialized data into memory (Or set page table entries so they can be faulted in) 4.Set up arguments on stack 5.Initialize registers (including SP, FP) 6.Jump to startup routine(Copies arguments to X0, ... and calls main and when main returns, do exit syscall)
353. **What are the actions that the handler goes through when an exception is handled?**	Read cause, and transfer to relevant handler; Determine action required; If restartable, Take corrective action & use EPC to return to program; Otherwise, Terminate program & Report error using EPC
354. **what are the pros/cons of a write-back?**	pro: good performance. con: difficult to implement
355. **what are the pros/cons of a write-through?**	pro: data is consistent between the memory and cache. con: is slow (solution: use a write buffer)
356. **What are the two types of branch addressing?**	B-type CB-type
357. **What determines how fast instructions are executed**	Processor and memory system
358. **What Determines how fast I/O operations are executed**	I/O system (including OS)
359. **what did the 80286 add, and when did it come out?**	24-bit addresses, MMU; 1982
360. **what did the i486 add and when did it come out?**	pipelined, on-chip caches and FPU; 1989
361. **what does a bit mask do**	Select some bits, clear others to 0
362. **what does abstraction do**	helps us deal with complexity by hiding lower-level detail
363. **What does a datapath do?**	performs operations on data
364. **what does an algorithm determine**	number of operations executed
365. **what does B.GE mean**	greater than or equal, signed
366. **what does B.GT mean**	greater than, signed
367. **what does B.HI mean**	greater than, unsigned
368. **what does B.HS mean**	greater than or equal, unsigned
369. **what does B.LE mean**	less than or equal, signed)
370. **what does B.LO mean**	less than, unsigned
371. **what does B.LS mean**	less than or equal, unsigned)
372. **what does B.LT mean**	less than, signed
373. **What does EM64T stand for?**	Extended Memory 64 Technology
374. **what does including bits in a word do?**	Set some bits to 1, leave others unchanged
375. **what does it mean to pipeline?**	Several multiplications performed in parallel
376. **what does java's just-in-time compiler do?**	Compiles bytecodes of "hot" methods into native code for host machine
377. **what does LDURB do?**	zero-extend loaded byte

378. **what does LDURSB do?**	sign-extend loaded byte
379. **what does linking object modules do?**	Produces an executable image
380. **what does MOVK do**	move with with keep (16 bits)
381. **what does MOVZ do**	move wide with zeros (16 bits)
382. **what does saturating operations mean?**	On overflow, result is largest representable value
383. **what does SIMD stand for?**	single-instruction, multiple-data
384. **what does system software do**	Compiler: translates HLL code to machine code
385. **what does the control contain**	sequences datapath, memory, etc.
386. **What does the following code do:** **B 1000**	go to location 10000ten
387. **what does the following code do:** **CBNZ X19, Exit**	go to Exit if X19 != 0
388. **what does the following code do:** **LDR X2, [X0,X1]!**	Adds X1 to X0 and stores the result in X0. Then uses that result as the address in main memory to fetch from
389. **what does the following code do:** **LDR X2, [X0,X1]**	This says add the contents of registers X0 and X1 and use the result as the address in main memory to fetch from
390. **what does the following code do:** **LDR X2, [X0,X1, LSL #3]**	Shift X1 left by 3 bits (so multiply by 8), then add that result to X0 and use the result as the address in main memory to fetch from
391. **what does the following code do:** **MOV X9,X0 // p = address of array[0]** **LSL X10,X1,#3 // X10 = size * 8** **ADD X11,X0,X10 // X11 = address of array[size]** **loop2: STUR XZR,0[X9,#0]// Memory[p] = 0** **ADDI X9,X9,#8 // p = p + 8** **CMP X9,X11 // compare p to &array[size]** **B.LT loop2 // if (p < &array[size]) go to loop2**	clears an array via pointers

392.	**what does the following code do:** **MOV X9,XZR // i = 0** **loop1: LSL X10,X9,#3 // X10 = i * 8** **ADD X11,X0,X10 // X11 = addressof array[i]** **STUR XZR,[X11,#0] // array[i] = 0** **ADDI X9,X9,#1 // i = i + 1** **CMP X9,X1 // compare i to size** **B.LT loop1 // if (i < size) go to loop1**	clears an array
393.	**what does the following code mean:** **B L1**	branch unconditionally to instruction labeled L1;
394.	**what does the following code mean:** **CBNZ register, L1**	if (register != 0) branch to instruction labeled L1;
395.	**what does the following code mean:** **CBZ register, L1**	if (register == 0) branch to instruction labeled L1;
396.	**what does this line mean?:** **mov $1, %rax**	system call 1 is write
397.	**what does this line mean?:** **mov $1, %rdi**	file handle 1 is stdout
398.	**what does this line refer to?:** **mov $13, %rdx**	number of bytes
399.	**what does this line refer to?:** **mov $message, %rsi**	address of string
400.	**what happens if divisor > dividend bits?**	0 bit in quotient, bring down next dividend bit
401.	**what happens If divisor ≤ dividend bits**	1 bit in quotient, subtract
402.	**what is a block in the cache?**	basic unit of cache storage; unit of copying. may be multiple words
403.	**what is a cache miss?**	The event when a memory access results in a memory location that is not in cache.
404.	**What is a fully associative cache?**	cache structure in which a block can be placed in any location in the cache
405.	**what is a Header?**	described contents of object module
406.	**What is a local area network (LAN)?**	Ethernet

407. **What is Amdahl's Law?**	gives a commonsense ceiling on performance (increased speed by a "better way" is limited by the usability of the "better way")
408. **What is Amdah's Law and its formula**	Improving an aspect of a computer and expecting a proportional improvement in overall performance is false. Given by: Timproved = (Taffected/improvement factor) + Tunaffected
409. **what is a multicore microprocessor**	More than one processor per chip
410. **what is AND**	what is bit-by-bit AND
411. **what is an instruction set**	The repertoire of instructions of a computer
412. **What is an N-way set associative cache?**	cache structure that consists of a number of sets, which consist of n blocks. each block in the memory maps to a unique set in the cache, and a block can be placed in any element of that set
413. **what is assembly language**	Textual representation of instructions
414. **what is a Static data segment?**	data allocated for the life of the program
415. **what is a Symbol table?**	global definitions and external refs
416. **what is a Text segment?**	translated instructions
417. **what is a TLB?**	translation lookaside buffer (TLB); buffer that memory management hardware uses to improve virtual address translation speed (NOT used in the cache)
418. **What is a wide area network (WAN)?**	the internet
419. **What is cache memory?**	Small fast SRAM memory for immediate access to data
420. **what is Clock frequency (rate)**	cycles per second
421. **what is Clock period**	duration of a clock cycle
422. **what is cpu clocking**	Operation of digital hardware governed by a constant-rate clock
423. **what is DRAM?**	Dynamic RAM, the most common form of memory that must be refreshed occasionally (data is stored as a charge in a capacitor)
424. **what is dynamic scheduling?**	when the CPU executes instructions out of order to avoid stalls
425. **what is EOR**	exclusive or
426. **what is hardware representation**	Binary digits (bits), Encoded instructions and data
427. **what is Immediate Post-indexed Addressing Mode**	just like immediate pre-indexed except the address in the base register is used to access memory first and then the constant is added or subtracted later.
428. **what is included in implementation**	The details underlying and interface
429. **what is Instruction set architecture**	The hardware/software interface
430. **What is Instruction set architecture (ISA)**	The hardware/software interface
431. **What is LRU page replacement?**	Least Recently Used
432. **what is LSL**	shift left

433. **what is LSR**	shift right
434. **what is multiple issue?**	a scheme whereby multiple instructions are launched in one clock cycle
435. **what is MVN**	Bit-by-bit NOT
436. **what is ORR**	bit-by-bit OR
437. **what is paralleling?**	using multiple resources to solve problems concurrently
438. **what is pipelining?**	an implementation technique in which multiple instructions are overlapped in execution
439. **what is Register Post-Indexed Addressing Mode**	The same Immediate Post-Indexed, except you add or subtract a register instead of a constant
440. **what is Register Pre-Indexed Addressing Mode**	The same as Immediate Pre-Indexed, except you add or subtract a register instead of a constant
441. **what is Relocation info?**	for contents that depend on absolute location of loaded program
442. **what is Response time**	How long it takes to do a task
443. **what is Scaled Register Pre-indexed Addressing Mode**	The same Register Pre-Indexed, except you shift the register before adding or subtracting it
444. **what is shamt**	shift amount
445. **What is sign extension?**	Representing a number using more bits. Preserves numeric value.
446. **what is SIMD?**	Single Instruction Multiple Data (the same instruction is applied to many data streams/AKA vector architecture)
447. **what is SISD?**	single instruction single data (a uniprocessor)
448. **what is SRAM?**	Static RAM, a lower-power and faster but more expensive type of memory, used for CPU caches
449. **what is the ARM instruction size?**	32 bits
450. **what is the benefit of having a larger cache block size?**	fewer cache misses
451. **what is the best performance measure?**	execution time
452. **what is the difference between a signed and unsigned bit**	For a signed integer one bit is used to indicate the sign - 1 for negative, zero for positive. Thus a 16 bit signed integer only has 15 bits for data whereas a 16 bit unsigned integer has all 16 bits available. This means unsigned integers can have a value twice as high as signed integers (but only positive values).
453. **what is the difference between a write-through and a write-back?**	for a write-through, when cache is updated, memory is also updated. for a write-back, memory is only updated when the block of cache is replaced
454. **what is the difference between the GPU and CPU?**	GPU's processing is highly data-parallel, the GPU doesn't have any branching/logic like the CPU, and the GPU has a very small cache/a lot less memory
455. **what is the downside to direct mapped cache?**	each block needs to use 2 bits to store the address

456. **what is the downside/upside to a write through vs a write back?**	downside: slow upside: memory is synchronized
457. **what is the equation for total time a task will take?**	# of instructions **# of clock cycles/instruction** clock cycle time
458. **What is the extended sign bit of the following:** **+2: 0000 0010 =>**	0000 00000000 0010
459. **What is the extended sign bit of the following:** **-2: 1111 1110 =>**	1111 11111111 1110
460. **What is the formula for determining address**	Address = PC + offset (from instruction)
461. **what is the formula for power in CMOS IC technology**	Power = Capacitative load x Voltage^2 x Frequency
462. **what is the goal of parallel computing?**	improve performance
463. **what is the Hamming SEC code?**	way to detect a parity code and make things more reliable
464. **What is the instruction to microoperation ratio of a complex instruction?**	1-many
465. **What is the instruction to microoperation ratio of a simple instruction?**	1-1
466. **what is the memory hierarchy?**	disk memory at bottom (cheapest, but slowest), then DRAM, then SRAM (cache), then registers
467. **what is the most popular embedded core?**	ARM
468. **what is the principle of locality?**	the tendency of a processor to access the same set of memory locations repetitively over a short period of time
469. **what is the purpose of Debug info?**	for associating with source code
470. **what is Throughputn**	Total work done per unit time
471. **what limits performance improvements**	power
472. **what makes up the Application binary interface**	The ISA plus system software interface
473. **what material are wafers made from?**	silicon inglot
474. **what size is the ARM address space?**	32-bit flat
475. **what sort of trade-off must be accepted with a faster multiply?**	cost/performance
476. **what tool helps blocks to be found quickly in virtual memory?**	translation lookaside buffer
477. **what two steps are involved in array indexing?**	1. Multiplying index by element size 2.Adding to array base address
478. **what type of hazard is the following?** **j tries to read a source before i writes it, so j incorrectly gets the old value.**	data hazard
479. **what type of hazard is the following?** **j tries to write a destination before it is read by i , so i incorrectly gets the new value.**	data hazard

480.	**what type of hazard is the following?** **j tries to write an operand before it is written by i. The writes end up being performed in the wrong order, leaving the value written by i rather than the value written by j in the destination.**	data hazard
481.	**what type of instruction does the compiler insert to produce a bubble?**	a nop
482.	**what was the 8080, and what year did it come out?**	8-bit microprocessor; 1974
483.	**what was the 8086, and what year did it come out?**	16-bit extension to 8080; 1978
484.	**what was the 8087, and what year did it come out?**	floating-point coprocessor; 1980
485.	**what was the 80386, and when did it come out?**	32-bit extension; 1985
486.	**What will Adding two +ve operands do?**	Overflow if result sign is 1
487.	**What will Adding two -ve operands do?**	Overflow if result sign is 0
488.	**when comparing performance, we say...**	"X is n times faster than Y"
489.	**when did AMD64 come out and what did it do?**	2003; extended architecture to 64 bits
490.	**when did the intel core come out?**	2006
491.	**when did the Pentium come out and what did it add?**	1993; superscalar, 64-bit datapath
492.	**when did the pentium II come out?**	1997
493.	**when did the pentium III come out?**	1999
494.	**when did the pentium IV come out?**	2001
495.	**when did the pentium pro come out?**	1995
496.	**when did we hit a physical limit on hardware?**	2005
497.	**when loading a program, we load from image file on disk into...**	memory
498.	**When you exclusive or (E0R) a register with itself, what happens?**	it zeroes out
499.	**x86 Instruction Encoding features ___ length encoding.**	variable