1. RISC-V makes it easy to decode the source registers by having them be in the same location for all instructions. This means that the decoding hardware can be simpler, faster, and cheaper.
2. RISC-V has 3 addressing modes: PC-relative, Register-offset, and Absolute
3. Some instruction types need longer immediates than others, and so they are not required to be in the same location for every instruction. They also do not need to be contiguous because their bits can simply be concatenated.
   1. The data path is made up of the elements that process data and addresses, such as registers, ALUs, multiplexors, and memories. The control path is made up of the element which process instructions and control flow, such as the branch unit and the instruction decoder. They usually flow together because data is usually processed in response to an instruction, and instructions often require the data to be processed in order to continue executing.
   2. A structural hazard is when a required resource is busy or unavailable. A data hazard is when one instruction needs to wait for the result of a previous instruction in order to finish executing. A control hazard is when the control flow of the program depends on the outcome of a previous instruction (e.g., a conditional branch)
   3. Code scheduling is when the instructions are reordered such that fewer instructions rely on a “load” result from the previous executed instruction. This is because “load”s can take multiple cycles to make their result available, and so without code scheduling, some instructions have to stall until the result is available.
   4. Forwarding paths are hardware shortcuts which allow the CPU to use the result of some calculations as soon as they are calculated rather than waiting for them to be stored in a register. They can mitigate data hazards and sometimes stalls.
   5. No, pipelines can have an arbitrary number of stages yielding diminishing returns when more and more subdivisions are added.
   6. Branches can cause bubbles because the pipeline needs to halt in order to wait for a computation to complete so as to know which instruction to fetch next. If bubbles are not added, then the wrong instruction might be fetched and perhaps start to execute, creating side-effects. They can be mitigated using forwarding.
   7. Branch prediction is a technique used to minimise the branch penalty by trying to dynamically predict which instruction needs to be fetched next. It involves a lookup table containing branches and whether or not they were taken. When the next branch appears, the CPU can assume that the same action will be taken next time. This allows the pipeline to start fetching the correct instruction earlier and more often.
   8. The cache miss rate is the proportion of the memory accesses for which the desired data is not in the cache, and so the data must instead be fetched from the next memory in the hierarchy. This needs to be low because a great penalty is incurred when there is a cache miss.
   9. Due to limitations on the speed of light, a fast memory can only be so big. Therefore, if more memory is needed, it either needs to be much slower, or it needs to have some fast parts and some slower parts. This is the hierarchy.
   10. The principle is that if an address in memory is accessed, it is likely that the addresses near it, or it itself, will be accessed again soon in the future. Therefore, storing the data therein in the cache will drastically speed up data access on average.
   11. Direct-mapped caches are ones in which each address can only correspond to one location in the cache. This makes for faster lookup times but can result in pollution and a high miss rate. Set-associative caches are a set of N direct-mapped caches in which lookups occur simultaneously. Fully associative caches are ones in which any address can be stored anywhere in the cache, and lookups occur simultaneously at every cache line. They can have a lower miss-rate than direct-mapped cache but a larger circuit area.
   12. A write-through policy is one in which a data-write hit results in not only the cache being updated, but the corresponding block in memory would be updated too. This keeps the cache consistent with the memory at all times but makes write operations take longer. Write-back policies on the other hand just update the cache but marks the cache block as “dirty” i.e., inconsistent with the memory. The memory will then be updated when the dirty block is replaced.
   13. Pipelining improves the clock frequency by increasing instruction throughput
   14. It can be necessary to stall the pipeline due to hazards. Sometimes an instruction will rely on the result of a previous instruction which has not finished executing yet, and so must stall until that result becomes available.
   15. The CPI of a pipelined processor is likely to be better in the aggregate because each clock cycle can include work towards multiple instructions but will likely be worse for each individual instruction because of stalls.
   16. We might encounter a structural hazard because two instructions which are being executed simultaneously in the pipeline might both need access to the same resource, and so one will have to stall to wait for the resource to become available.
   17. Branches that branch backwards are likely to be part of a loop, and loops by their very nature often loop backwards many times, but only exit once. Therefore, in a loop the backwards path is usually much more common than the forwards path.
   18. The compiler could organise the instructions in such a way that the number of instances in which an instruction relies on the result of the previous instruction is minimised.
   19. You need to have registers in between each stage of the pipeline, and so more time and hardware would be used to simply move data around, making the performance gain negligible.
4. // TODO
   1. More data can be stored in the cache and so it is more likely that the desired data is in the cache.
   2. Larger cache blocks mean that for a fixed-size cache, fewer blocks can be stored in the cache.
   3. The benefit of caches would be much less significant if addresses were accessed randomly because caches rely heavily on assumptions of locality.