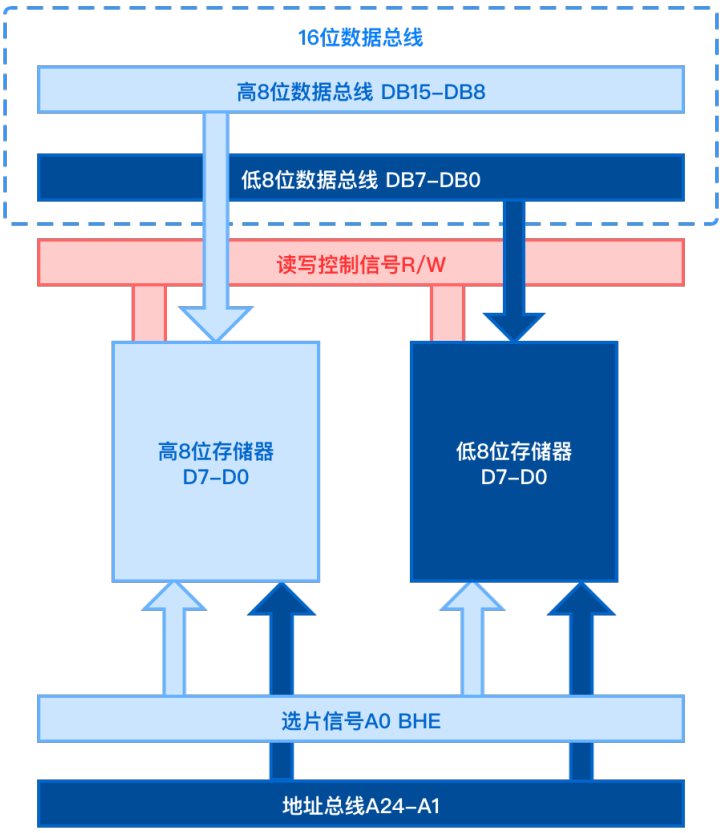
2.6

地址空间大小为 $2^{24} = 16$MB, 存储器组织如下图:



使用两块存储器分别用于存储 16 位中的高 8 位和低 8 位; 剩下的 23 位用来选择地址. 如果需要操作完整的 16 位, 则两片存储器同时操作; 否则可以选择其中的一片来完成高 8 位或者低 8 位的读取.

2.10

微指令指的是面向微程序级别的指令, 用来描述微操作 (相对于机器指令被机器直接读取、宏指令面向软件).

一条 (机器) 指令对应了一个微程序, 而一个微程序由多条微指令组成.

在基于微程序设计的控制器中, 每个微程序 (其包括的微指令) 都存放在控制 ROM 中, 并且有唯一的 (微) 地址. 译码器读取到指令操作码后开始在控制 ROM 中寻址, 按顺序读出每一条微指令, 产生控制信号, 来执行微程序.

2.13

①程序计数器内容 0x20000004 送到地址生成部件, 通过地址总线、到地址译码器进行译码, 寻址这条指令.
②操作控制器从 0x20000004 读出内容“LDR R1,[R3]”, 经过数据总线存入到指令寄存器 IR 中.
③程序计数器自增, 准备读取下一条指令.
④指令译码器 ID 对操作数译码, 操作控制器 OC 按时序发出相应控制信号.
⑤R1 为目的操作数,[R3] 为源操作数,[] 表示间接读取存储器中的内容.
⑥控制信号按顺序执行, 控制器在内存中寻找 R3 寄存器中存放地址, 读取其中内容, 存放到 R1 中.

2.14

不能.

为了叙述方便, 不妨设 a 位存储器、b 位地址总线,a<b.

由于存储器单元都有 b 位的地址, 在定长指令中无法用 a 位的地址码给出 b 位的源和目标地址, 因此需要不止一个指令周期的寻址操作来进行寻址.

2.15

①WAW(Write After Write)

I1: MOV R1,#00000001

I2: ADD R1,R1,#00000002

如果 I2 的写入在 I1 之前执行, 那么结果 R1 里存的就是 I1 的结果 #1. 但正确的逻辑结果应该是 I2 的 ADD 操作后的 #3.

②WAR(Write After Read)

I0: MOV R1,#00000001

I1: STR R1,[R3]

I2: MOV R1,#00000002

如果 I2 的写入先于 I1 的读取, 那么存入 R3 的数据就是 I2 中的 #2, 但是程序的正确逻辑结果应该是 I0 中的 #1.

2.16

①转移指令跳转到的目标指令地址 I_k

②假设从 I_j 跳转到 I_k, 则在流水线上, 处理 I_j 时多需要三个周期, 会导致 I_{j+1} 和 I_{j+2} 被装载到流水线上, 造成两个流水线周期延迟.

③转移指令后面一个时间片

④转移目标缓冲器, 收集储存近期所有转移信息并打表; 转移时查表进行操作

2.18

原因: 必须保证每条流水线上的指令不相干.

例如:

I1: MOV R1,#00000001

I2: ADD R1,R1,#00000002

两条指令用到牵扯到对同一个寄存器的改写, 因此不能在不同的流水线上并行.

2.20

①同构多核处理器指多核的每一个内核采用相同的结构、地位相等, 简单增加处理器数量.

②异构多核处理器指配置不同功能和性能的内核来匹配实际应用需求, 提升芯片总体性能的同时优化处理器结构、降低系统功耗.

举例: 通过将 CPU 和 GPU 集成到一颗芯片上, 形成异构多核, 串行执行部分在 CPU 上处理, 并行部分在 GPU 上处理, 以此来提升处理速度.

2.24

1.1MIPS/MHz × 300MHz=330MIPS

2.25

不能. 例如, 对于 RISC 和 CISC, 相同的任务下前者花费指令更多, 因此 MIPS 不能简单地作为唯一评判指标.