第三章作业参考答案

2. 解：(1) 主存的地址码为 26 位，因此总的存储单元数目为 2^{26}，每个存储单元大小为 64 位。现采用内存条的形式，每条内存条的存储单元为 $16M=2^4\times2^{20}=2^{24}$，每个存储单元大小为 64 位，因此每个内存条所需的地址码长度为 24 位。总共的存储单元为 2^{26}，每内存条的存储单元为 2^{24}，故所需的内存条数目为：

$$\frac{2^{26}}{2^{24}}=4\left(\text{条}\right)$$

(2) 现在已有的 DRAM 存储芯片为 4M×8 位，因此每块内存条需要用这些 DRAM 来进行字长及容量的扩充，所需的芯片数为：

$$\frac{16\text{M}\times64}{4\text{M}\times8}=32\left(\text{片}\right)$$

(3) 主存需要 4 个内存条，每个内存条需要 32 片 DRAM 芯片，因此共需的 DRAM 芯片为：

$$4\times32=128\left(\text{片}\right)$$

如（1）中分析，每个内存条需要 24 根地址线(A23~A0)完成内存条内存储单元寻址。现一共有 26 根地址线，因此剩下的 2 根地址线可以用来选中不同的内存条。例如，可以采用 2 根高位地址线(A25~A24)，通过 2-4 译码器译码产生片选信号对各模块板进行选择。

6. 解：（1）组成的只读存储器为 128K×16，即总共有 128K 个单元，每个单元存储 16 位。也就是说每次读出 16 位，因此数据寄存器的位数为 16。

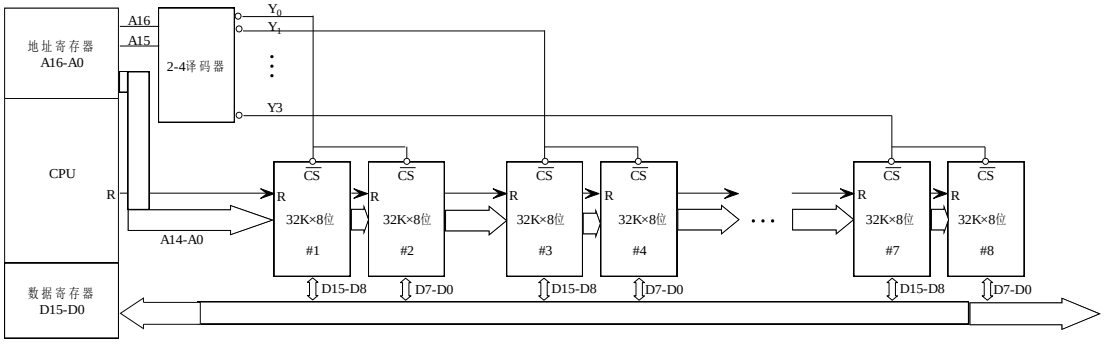
（2）只读存储器有 128K 个单元，因此需要 17 根地址线，即地址寄存器需要 17 位。

（3）所需的存储芯片的数量为：

$$\frac{128\text{K}\times16}{32\text{K}\times8}=8\left(\text{片}\right)$$

这 8 片 E^2PROM 芯片分成 4 组，每组都是 32K×16 位。

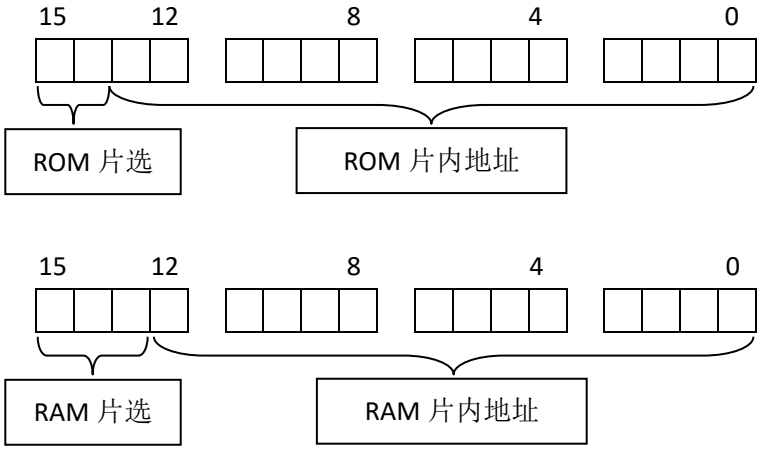
（4）由上可知，存储器的总地址线为 17 根，片内所需的地址线为 15 根。假定片内用低 15 位地址 A14-A0；高 2 位地址 A16-A15 则用于片选，经 2-4 译码后选中 4 组 SRAM 芯片。因此，存储器的组成框图如下：



7. 解：（1）计算芯片数量。ROM 不需要扩充；RAM 则需采用 8K×8 位的 RAM 芯片进行容量扩充至 40K×16 位，因此所需的 RAM 芯片为 10 片，分成 5 组，每组容量为 8K×16 位。

（2）分析地址线。ROM 的地址空间为 0000~3FFFH，即 ROM 的单元个数共为 4000H 个。其二进制形式等价为：最高 2 位为 01，其余 14 个比特都等于 0。再转换成十进制，即总容量为 2^{14}=16K。RAM 的容量为 40K，这样 ROM 和 RAM 的总容量为 16K+40K=56K，因此所需的最小地址线长度为 16 位。

对于 ROM 而言，片内需要 14 根地址线，余下的 2 根地址线可以用作片选；对于 RAM 而言，所采用的 RAM 芯片是 8K×16 位的，故片内需要 13 根地址线，余下的 3 根地址线用作片选。ROM 和 RAM 的地址结构如下图所示。但这样一来，ROM 和 RAM 的片选线数量不一致，下面通过分析说明解决方法。



根据题意，ROM 的地址空间为 0000~3FFFH。把这些十六进制地址转换成二进制，并按照上图中 ROM 的地址格式进行分析，可知 ROM 的所有地址的片选位 A15-A14 都为 00。例如，对于地址 0000H，二进制形式为 0000 0000 0000 0000B，其中 B 代表二进制，因此 A15-A14 的最高 2 位为 00；对于地址 3FFFH，二进制形式为 0011 1111 1111 1111B，因此 A15-A14 的最高 2 位也为 00。所有地址都这样进行分析，就很容易知道：对应 ROM 而言，A15-A14 必须为 00。

根据题意，RAM 的起始地址为 6000H。转换为二进制形式 0110 0000 0000 0000B，再结合上图中 RAM 的地址结构可知，RAM 的片选为 011。类似地分析 RAM 的所有地址范围 6000~FFFFH，可知：

- ✧ 对于地址范围 6000~7FFFH，RAM 的片选位均为 011；
- ✧ 对于地址范围 8000~9FFFH，RAM 的片选位均为 100；
- ✧ 对于地址范围 A000~BFFFH，RAM 的片选位均为 101；
- ✧ 对于地址范围 C000~DFFFH，RAM 的片选位均为 110；
- ✧ 对于地址范围 E000~FFFFH，RAM 的片选位均为 111。

上述的每个地址范围，容量都是 2000H 个单元（即 8K 个单元），因此总容量刚好是 40K，满足题目要求。也就是说，RAM 的片选位 A15-A13 从 011 开始往上增长，如 011、100、……、111。

由上分析可知，ROM 与 RAM 的片选位数不一致，ROM 的片选位为两位，对应数值为 00；RAM 的片选位为 3 位，对应数值从 011 逐一增大至 111。若分开译码，则需要设计 2 个译码器，一个为 2-4 译码器，一个为 3-8 译码器。但这样的话，2-4 译码器只用其中一个对应 00 的 1 个译码片选线；3-8 译码器只用其中对应 011~111 的 5 个译码片选线。因此，

这两个译码器都将有 3 个译码输出端用不上，这样会造成浪费，增加设计成本。为简化设计，我们可以统一使用高 3 位地址线（A15-A13）当作片选线，并经 3-8 译码器产生 8 条片选线 Y0、Y1、……、Y7。由于 ROM 的真正片选线是 A15-A14，A13 是用于片内地址的，为简化设计才加进 3-8 译码器的，因此 A15-A14 相同而 A13 为 0 和 1 时对应的译码器输出片选线都需用作 ROM 的片选。结合前面的分析可知，A15-A14 为 00 而 A13 为 0 和 1 时，3-8 译码器对应的 2 条片选线可用作 ROM 的片选信号；而 A15-A13 为 011-111 时，3-8 译码器对应的 5 条片选线可以用作 RAM 的片选信号，分别对应 5 组芯片。A15-A13 为 010 时，对应的片选线则未作使用，可以留给日后扩充用。

综合上面的分析，可知各 ROM 和 RAM 芯片的地址译码方案如下图所示，左边给出的十六进制数是每个模块的起始地址。

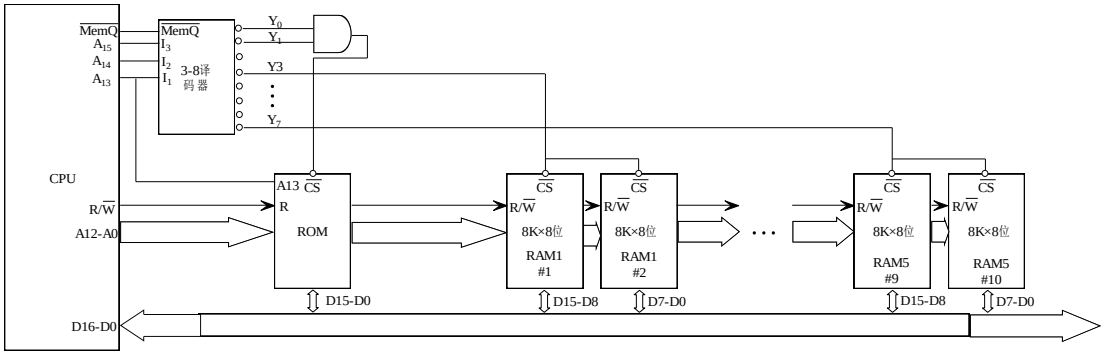
0000 H	ROM
4000 H	未用
6000 H	RAM_1
8000 H	RAM_2
A000 H	RAM_3
C000 H	RAM_4
E000 H	RAM_5

（2）由前面的分析可知，A15-A13 为 000 和 001 时， 3-8 译码器对应的输出片选线 Y0 和 Y1 用于选中 ROM。但 ROM 的片选信号 $\overline{CS}$ 只有一个，因此不能同时将 Y0 和 Y1 连接到 $\overline{CS}$。根据译码器的原理（假定输出为低电平有效）可知，当 A15-A13 为 000 时，Y0 为 0，而 Y1-Y7 都为 1；当 A15-A13 为 001 时，Y1 为 0，而其它都为 1。也就是说，用于选中 ROM 的 Y0 和 Y1，要么是 0 和 1，要么是 1 和 0。进一步考虑到 ROM 的片选信号是低电平有效，因此 Y0 和 Y1 组合后的结果必须为 0，这显然可以用一个与门或者同或门来连接 Y0 和 Y1。即将 Y0 和 Y1 当作与门的输入，然后将与门的输出连接至 ROM 的 $\overline{CS}$ 端，见下面的逻辑框图。

题目尚要求用 $\overline{MemQ}$ 来控制存储器的访问，即当此控制信号为低电平时才能有效地访问包括 ROM 和 RAM 的存储器。具体实现时，可以考虑两个方案：（1）每一个存储芯片都连接这个控制信号；（2）直接用这个控制信号区控制 3-8 译码器。权衡利弊，显然后者的实施方案比较简便，因此将访存控制信号 $\overline{MemQ}$ 连接至 3-8 译码器，当它为低电平的时候才能选中 ROM 或 RAM 的芯片，才能进行读或写。

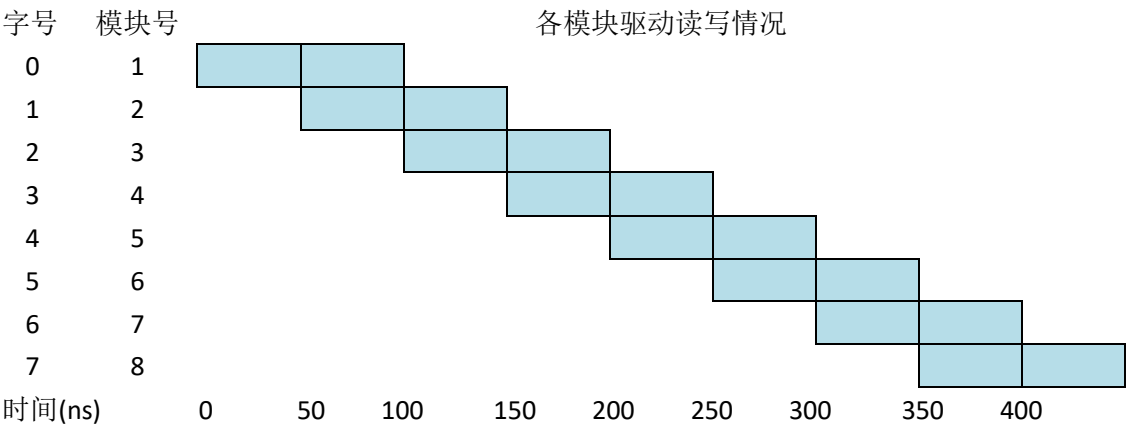
由于每个 ROM 或 RAM 存储芯片都有 R/$\overline{W}$ 控制信号，因此需要将 CPU 发出的 R/$\overline{W}$ 控制信号连接至每一个存储芯片。

根据上述的分析，可以得到如下存储器连接示意图。注意，由于 ROM 芯片的地址范围为 0000~3FFFH，因此片内是需要 14 根地址线的，这就需要将 A13 再连接到 ROM 芯片。



8. 解：（背景知识）存储器容量是 64M，存储模块有 8 个，因此每个存储模块的容量是 8M。对于顺序存储，第 1 个模块中所存储的字地址范围为 0 ~ 8M-1，第 2 个模块为 8M ~ 16M-1，依次类推。对于交叉存储，第 1 个模块存放的是那些字地址模 8 后 0 的那些字，第 2 个则为模 8 后为 1 的那些字，如此类推。

现假定连续读出 8 个字。顺序存储方式只能从第 1 个模块读出，读出时只能是前一个字完全读出后，才能给出下一个字的地址以读出下一个字，这样前后两个字读出的时间间隔就是存储周期 T=100ns。交叉存储方式可以从第 1 个模块读出第 0 个字，从第 2 个模块读出第 1 个字，如此类推。根据题意，存储周期为 T=100ns，即从 CPU 发出存储器读写的相关信号到能稳定地读出某一个字的时间为 100ns；总线周期为 τ =50ns，即从发出存储器读写的相关信号到某一个模块开始正式读某一个字的时间为 50ns，其余的 50ns 是模块稳定地读出该字的时间。因此，对于交叉方式，8 个模块读写的时间示意图如下。由图可知，CPU 实际上只能连续驱动 2 个模块同时读写，当第 2 个模块开始从存储模块中读数据时，前面的所有模块都已停止读数据而处于空闲状态了。也就是说，虽然有 8 个模块，但不能同时都工作。



题目并未提读多少个字，下面以读 m=8 个字为例，分别计算顺序存储和交叉存储的带宽。

（1）计算读出的数据位数。顺序存储器和交叉存储器连续读出 m = 8 个字的信息总量都是：

$$q = 64 \times 8 = 512 \text{ (位)}$$

（2）顺序存储器和交叉存储器连续读出 8 个字所需的时间分别是：

✧ 顺序存储读出一个字需要一个存储周期，8 个字则需 8 个存储周期，因此

$$t_1 = mT = 8 \times 100 = 800(ns) = 8 \times 10^{-7}(s)$$

✧ 交叉存储读出第一个字需要一个存储周期，后面的每一个字读出只需要一个总线传送周期，因此 8 个字所需的时间为：

$$t_2 = T + (m-1)\tau = 100 + (8-1) \times 50 = 450(ns) = 4.5 \times 10^{-7}(s)$$

（3）顺序存储器和交叉存储器的带宽分别为：

✧ 顺序方式：$W_1 = \dfrac{q}{t_1} = \dfrac{512}{8 \times 10^{-7}} = 64 \times 10^7 \left(b/\mathrm{s}\right)$ (位/秒)

✧ 交叉方式：$W_1 = \dfrac{q}{t_1} = \dfrac{512}{4.5 \times 10^{-7}} = 113.8 \times 10^7 \left(b/\mathrm{s}\right)$

9. 解：首先计算 cache 的命中率，为：

$$H=\frac{N_c}{N_c+N_m}=\frac{2420}{2420+80}=0.968$$

再计算 Cache/主存系统的平均访存时间，为：

$$T_a=HT_c+(1-H)T_m=0.968\times 40+(1-0.968)\times 240=46.4(ns)$$

因此，Cache/主存系统的效率为：

$$e=\frac{T_c}{T_a}\times 100\%=\frac{40}{46.4}\times 100\%=86.2\%$$

10. 解：根据平均访存时间计算公式$T_a=HT_c+(1-H)T_m$，可解得命中率为：

$$T_a=\frac{T_m\text{-}T_a}{T_m\text{-}T_c}$$

代入相关数值得到：

$$T_a=\frac{T_m\text{-}T_a}{T_m\text{-}T_c}=\frac{200\text{-}50}{200\text{-}40}=93.75\%$$

11. （选做）解：假设总线传送周期为τ，内存读/写周期为T_m，指令的执行时间为T_e。对于这两段循环程序的运行时间，要考虑 CPU 和存储器的联合工作过程，分析如下。

为方便比较，假定这两段程序运行时，都不考虑进程的调度和切换时间，只考虑该程序在 CPU 中运行的总时间，即包括从内存取指令及在 CPU 中执行的时间。这是比较的前提，下面分 CPU 中（或外）是否有 cache 来进行考虑。

（1）假设 CPU 中或外都没有 cache，CPU 为串行执行，即先利用程序计数器 PC 取出指令，然后分析指令并进行执行。执行完当前指令后，再根据 PC 取下一条指令，如是循环。根据 CPU 的工作原理，可以假定取指时间就是内存的读/写周期T_m，分析指令和执行的总时间就是前述定义的T_e。

在执行程序时，先取第一条指令，耗时T_m；然后进行执行，耗时T_e。执行完第一条指令后，已经耗时$T_m+T_e\left(>T_m\right)$，这时再根据下一条指令的 PC 值发出第二条指令的地址给内存。显然，在这种情况下，驱动内存的第一个模块和第二个模块的时间间隔已经超过T_m，内存的四体交叉存储器根本没法发挥其以流水方式进行读/写的作用。也就是说，不管是 6 条还是 8 条指令的循环程序，每条指令都需要耗时T_m+T_e，因此在执行的指令总量相同的情况下，各自的运行时间是相等的。

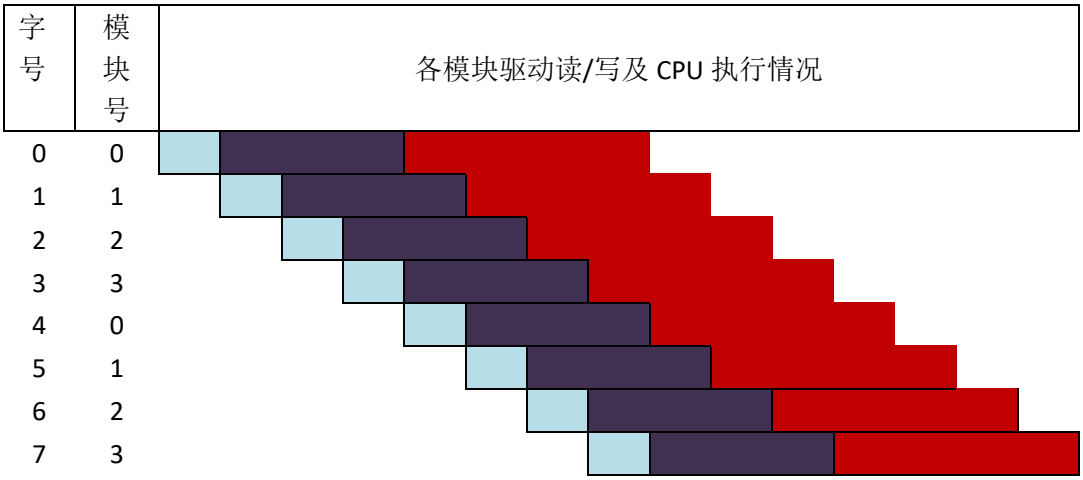
（2）假设 CPU 内或外部有 cache，为简化分析进一步假设 cache 的每一个块都刚好能够装 8 条指令。CPU 还是按照顺序方式执行，即先取指，然后再分析和执行。

在执行时，先取第一条指令。这时不可能在指令 cache 中命中，而是需要访问主存。在主存中找到该指令后，把该指令及其对应的指令块装入 cache 中。假设该 6 条或 8 条指令能够划分在同一块中，因此当 CPU 取第一条指令时，即能够把比较程序中的 6 或 8 条指令全部读入 cache 中。不管是 6 还是 8 条指令的程序，皆是以一块 8 指令的方式装进 cache 中。这 8 条指令是按地址连续的方式存放的，因此读内存时可以用流水方式进行读。也就是说，不管是 6 还是 8 条指令的循环程序，装入 cache 所需的时间为：

$$t_{cache}=T_m+7\times\tau$$

这 8 条指令装入 cache 及在 CPU 中执行的过程如下图所示，其中淡蓝色格子表示总线传送周

期，淡紫色格子代表存储芯片在得到地址及读信号后读出数据所需时间，暗红色格子代表在取出执行后 CPU 进行分析和执行所需的时间。



如图可以看出，这 8 条指令在装入 cache 的过程及 CPU 执行的过程按流水方式组织。因此，对于 6 条指令的循环程序，后面 2 条不是本循环程序所需的指令，因此后面两条的装入时间可以忽略。也就是说，完成这 6 条指令的第一轮循环所需时间为：

$$t_{round1}^{\text{Ins-6}}=(T_m+T_e)+5\tau$$

对于 8 条指令的循环程序，完成第一轮 8 条指令循环所需时间为：

$$t_{round1}^{\text{Ins-8}}=(T_m+T_e)+7\tau$$

在完成第一轮循环后，所有指令都已经保存在 cache 中，且假定没有被替换出去，则从第二轮循环开始，所有指令的取指都可以在 cache 中命中。假设从 CPU 发出指令地址，到 cache 命中后把所需指令调入 CPU，总共所需的时间为T_c。那么，再执行 79 轮 6 条指令的循环程序，所需时间为：

$$t_{round2\text{-}80}^{\text{Ins-6}}=(T_c+T_e)\times 79\times 6=474\times(T_c+T_e)$$

再执行 59 轮 8 条指令的循环程序，所需时间为：

$$t_{round2\text{-}60}^{\text{Ins-8}}=(T_c+T_e)\times 59\times 8=472\times(T_c+T_e)$$

综上可知，执行 6 条指令的循环程序，总共所需时间为：

$$t^{\text{Ins-6}}=t_{round1}^{\text{Ins-6}}+t_{round2\text{-}80}^{\text{Ins-6}}=(T_m+T_e)+5\tau+474\times(T_c+T_e)$$

执行 8 条指令的循环程序，总共所需时间为：

$$t^{\text{Ins-8}}=t_{round1}^{\text{Ins-8}}+t_{round2\text{-}60}^{\text{Ins-8}}=(T_m+T_e)+7\tau+472\times(T_c+T_e)$$

对比一下可知，除非$\tau=T_c+T_e$，否则这两个循环程序的运行时间不相等。

（3）其它情况，各人自己考虑。如可以考虑若 CPU 为流水 CPU，取指和执行按照流水方式进行，则结果又可能不一样。

13. 解：（背景知识）在 Cache/主存系统中，均需要对 Cache 和主存划分块，且两者的块大小都要相同。对于 Cache 而言，其块常称为“行”；对主存而言，其块还是称呼为“块”。

根据题意，Cache 共有 64 行，每组 4 行，因此总共有 16 组。主存有 4K 块，每块 128 字，总容量为 $4K\times128=2^{12}\times2^{7}=2^{19}$（字）。也就是说，总的地址长度为 19。

根据教材图 3.35，组相联映射时的内存地址结构为：

tag	组号	字号

根据上述的分析可知，这 3 个字段的总长度就是内存的地址长度，即为 19。字号用于为块内的字寻址，现每块有 128 字，因此字号长度为 7。组号标记 Cache 的组，现有 16 组，因此组号长度为 4。这样，剩下的 tag 长度就为 19-7-4=8 位。所以，此组相联映射时的内存地址结构为：

tag	组号	字号
8 位	4 位	7 位

29. 解：小端模式时，数据的低有效位放在低内存地址中；大端模式时，则将数据的高有效位放在低内存地址中。据此可得：

（1）小端模式时：0x6000 和 0x6001 分别放置 0x34 和 0x12；

（2）大端模式时：0x6000 和 0x6001 分别放置 0x12 和 0x34。