实验三 存储器实验

一、实验目的

1、了解静态随机存储器 RAM 和只读存储器 ROM 的工作特性及读写方法。

2、掌握存储器与总线的连接及存储器地址空间映射的原理。

二、实验内容

设计一个 8 位字长的存储器通路，包括 ROM 和 RAM 两个地址相互独立的存储器，实现对 ROM 和 RAM 存储器的数据读写操作及数据成批导入 ROM 的操作。

三、实验器件

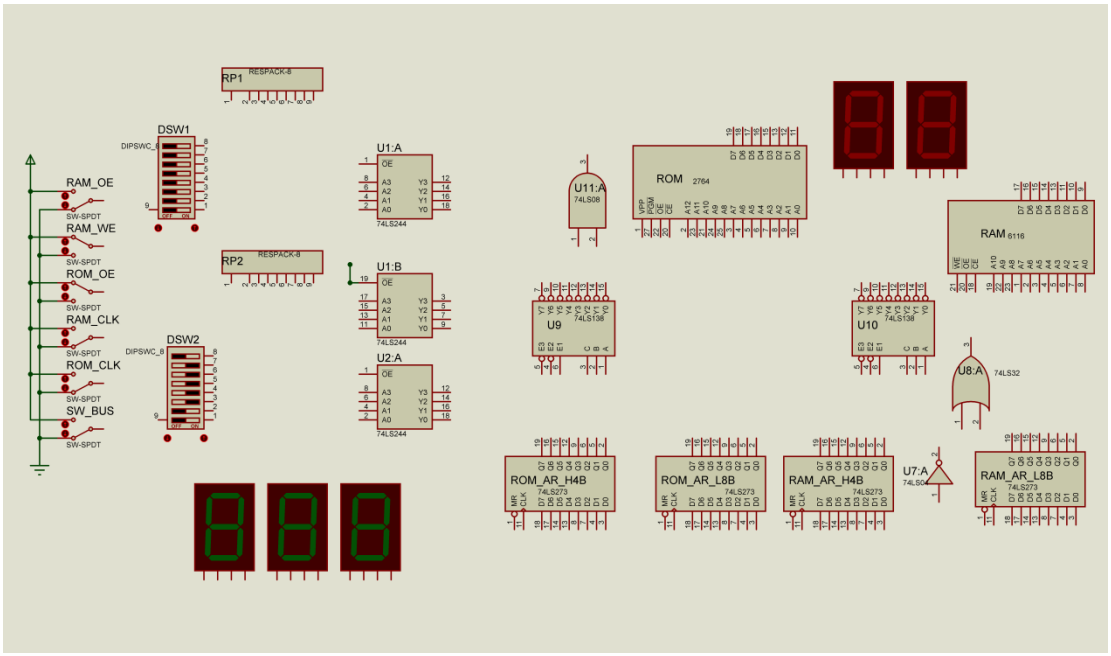
1、只读存储器 2764 及静态随机存储器 6116。

2、三态门（74LS244）、寄存器（74LS273）及 3-8 译码器(74LS138)。

四、实验原理

实验器件图如下所示：由地址输入单元、存储器及地址选择电路组成。

存储器通路共有两条总线：12 位地址总线 和 8 位数据总线。下图左边是拨码开关构成的 12 位地址输入端连在地址总线上，通过三个绿色数码管输出显示 12 地址信息。右边则是存储器 ROM、RAM 及其地址选择电路。ROM 和 RAM 存储器内部有三态门结构，其数据输出端直接连在数据总线上，通过两个红色数码管显示 8 位数据信息。



存储器是用来存储信息的部件，是计算机的重要组成部分，常见的半导体存储器类型主要有 ROM 和 RAM：ROM 是 Read Only Memory（只读存储器）的缩写，RAM 是 Random Access Memory（随机存取存储器）的缩写。ROM 存储器一般容量较大，在系统停止供电的时候仍然可以保持数据；ROM 只能读出数据，不能写入数据。而 RAM 存

储器一般容量较小，在系统掉电之后就丢失数据；RAM 即可读出数据，又可写入数据。本实验中使用的ROM 存储器型号是 2764（8K×8 位），RAM 存储器型号是 6116（2K×8 位）。

ROM 芯片 2764 的数据线 D0-D7 接到数据总线，地址线 A0-A8 由地址锁存器 74LS273 给出，用来对 ROM 片内存储单元寻址。其余地址线 A9-A12 接地，2764 有两个控制端： $\overline{CE}$ （片选）、 $\overline{OE}$ （读），都是低电平有效。

RAM 芯片 6116 的数据线 D0-D7 接到数据总线，地址线 A0-A7 由地址锁存器 74LS273 给出，用来对 RAM 片内存储单元寻址。其余地址线 A8-A10 接地。6116 有三个控制端： $\overline{CE}$ （片选）、 $\overline{OE}$ （读）、 $\overline{WE}$ （写），都是低电平有效。

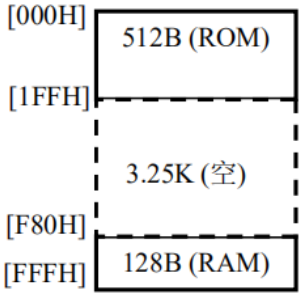
存储器通路控制信号的逻辑功能如下表 2-9 所示。值得注意的是：在对 ROM 或 RAM 读写的时候，首先必须在存储器的片选有效（$\overline{CE}$=0）前提下，才能对相应的存储器读（$\overline{OE}$=0）或写（$\overline{WE}$=0），例如：对 ROM 芯片 2764 进行读操作，必须使能$\overline{ROM_CE}$=0 且 $\overline{ROM\_OE}$=0，存储器片选信号$\overline{ROM_CE}$和$\overline{RAM_CE}$是由地址信号的高 4 位地址总线 8-11 经过**片选逻辑电路**自动形成，不需要拨码开关控制。

表 2-9 存储器通路控制信号说明

信号名称	作 用	有效电平
ROM_CLK	2764 地址的锁存脉冲信号	上升沿跳变有效（开关手动）
RAM_CLK	6116 地址的锁存脉冲信号	上升沿跳变有效（开关手动）
ROM_CE	2764 的片选有效信号	低电平有效 （地址驱动）
RAM_CE	6116 的片选有效信号	低电平有效 （地址驱动）
ROM_OE	2764 的读允许信号	低电平有效（开关手动）
RAM_OE	6116 的读允许信号	低电平有效（开关手动）
RAM_WE	6116 的写允许信号	低电平有效（开关手动）
SW_BUS	地址总线输入允许信号	低电平有效（开关手动）

其次，必须在地址锁存器（74LS273）ROM_AR、RAM_AR 锁存地址信号，才能选中存储器片内相应的单元。地址锁存器 ROM_AR 和 RAM_AR 的输入都连接至地址总线 0-11，在其 CLK 端开关出现上升沿跳变的时候，地址总线 0-11 的数据打入 ROM_AR 或 RAM_AR 锁存。锁存后无论地址总线如何变化，选中的存储单元也不会发生改变，可以进行稳定的读写操作（存储器数据端输入或输出）。存储器通路设计的最重要环节是存储器与地址总线的连接，因为连接方式决定了存储器地址空间的映射关系，即决定了每个存储器芯片在整个存储空间中的地址范围。12 位地址总线的理论地址空间为 4K（000H-FFFH），本实验分配其中最低的 512 地址为 ROM 区 (000H-1FFH)，最高的 128 地址为 RAM（F80H-FFFH），其余留空，如下表 2-10 所示。

表 2-10 存储器与地址总线连接方式及地址空间范围



A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 …… 0 0 0 1	0 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1
……	……
1 1 1 1 …… 1 1 1 1	1 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1

存储器通路设计一般将地址总线区分为低位地址线和高位地址线两部分：低位地址线直接和存储器芯片的地址信号连接作为片内地址译码，而高位地址线的连接主要用来产生片选信号（称为片间地址译码），以决定每个芯片在整个存储系统中的地址范围。在本实验中，12 位理论地址空间分为低 8 位地址线和高 4 位地址线。低 8 位地址线 0-7 分别与 ROM 和 RAM 芯片的地址线 A0-A7 共用；高 4 位地址线 8-11 则通过两个 3-8 译码器把地址空间分为 16 个部分。ROM 芯片的片选由低位 3-8 译码器的最低 2 个部分片选信号的片选电路形成（**思考为什么？**），RAM 芯片的片选由高位 3-8 译码器 74LS138 的最高 1 个部分片选信号与 RAM 的 A7 地址线的片选电路形成（**思考为什么？**）。值得注意的是：片选电路的逻辑组合不是唯一的，可以有多种实现形式。片选电路中采用的译码器是把输入二进制代码译成相应状态输出的器件，一般有 2-4 译码器（74LS139）、3-8 译码器(74LS138)等规格。而 4-16 线译码功能则可以采用两个 3-8 译码器通过片选信号组合实现。本实验使用的译码器 74LS138 的逻辑功能如下表 2-11 所示：

表 2-11 3-8 译码器 74LS138 逻辑功能表

输入						输出							
E_1	$\overline{E}_2$	$\overline{E}_3$	A_2	A_1	A_0	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	1/×	×/1	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

注：A0、A1、A2 分别对应 A、B、C,根据这个表，ROM 和 RAM 的地址空间，连接锁存器和 74LS138 等器件，完成 ROM 和 RAM 的片选逻辑电路。

ROM 批量导入数据的技巧

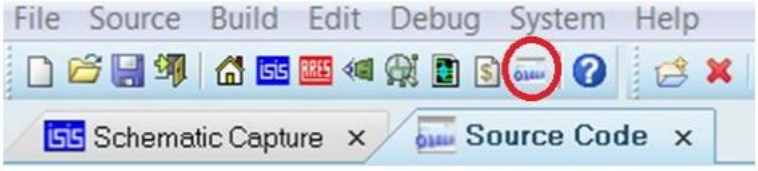
把一组程序或数据一次性批量导入 ROM 中，才能使 ROM 在往后的实验中可以充当程序存储器或数据存储器的角色。通过使用 Proteus 的 8051 汇编器中的伪汇编指令来实现上述功能。具体操作步骤如下：

1）新建 txt 文件，把.txt 后缀改名为.asm 后缀，然后按照下列伪汇编指令格式输入数据：

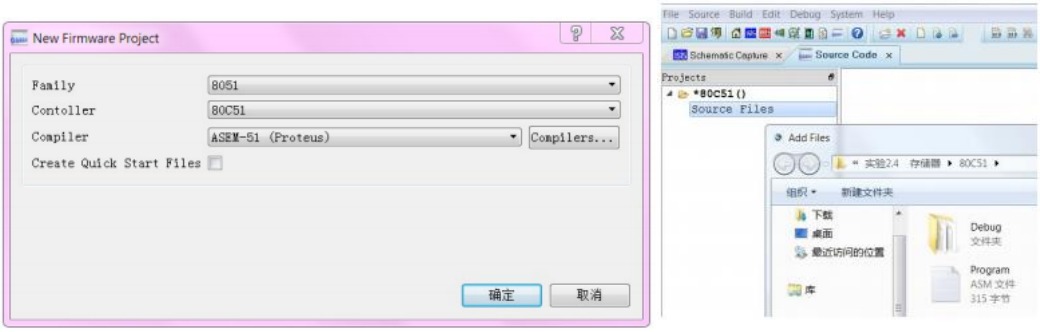
```
ORG 0024H
    DB 01010101B
    DB 01010101B
    DB 01010101B
    DB 01010101B

    DB 01H
    DB 01H
    DB 01H
    DB 01H

    DB 0FFH
    DB 0FFH
    DB 0FFH
    DB 0FFH

    DB 01H
    DB 01H
    DB 01H
    DB 01H
END
```

注：上述代码中 的语句“ORG xxxxH”规定该语句后所跟数组存储的首地址，数组末尾必须以其他 ORG 语句或“END”作为结束。在 asm 文件中可以使用多个“ORG”语句来规定在存储器的不同位置存放不同长度的数组。在“ORG”语句后的“DB xxxxxxxxB”或“DB xxH”语 句都表示一个存储单元存放的 8 位数据，前者是二进制，x 表示 0 或 1；后者是十六进制，x 表示 0-9 和 A-F（若 x=A-F，则要写成 0AxH-0FxH）。注意：如果已经通过 ORG 语句定义的数据段需要清零，需要写一段全零的 ORG 语句来覆盖，否则数据永远存在。

2）Proteus 仿真环境的菜单栏里点击 Source Code 图标（红色圈标注的地方），弹出 Source Code 标签页，如下图所示：

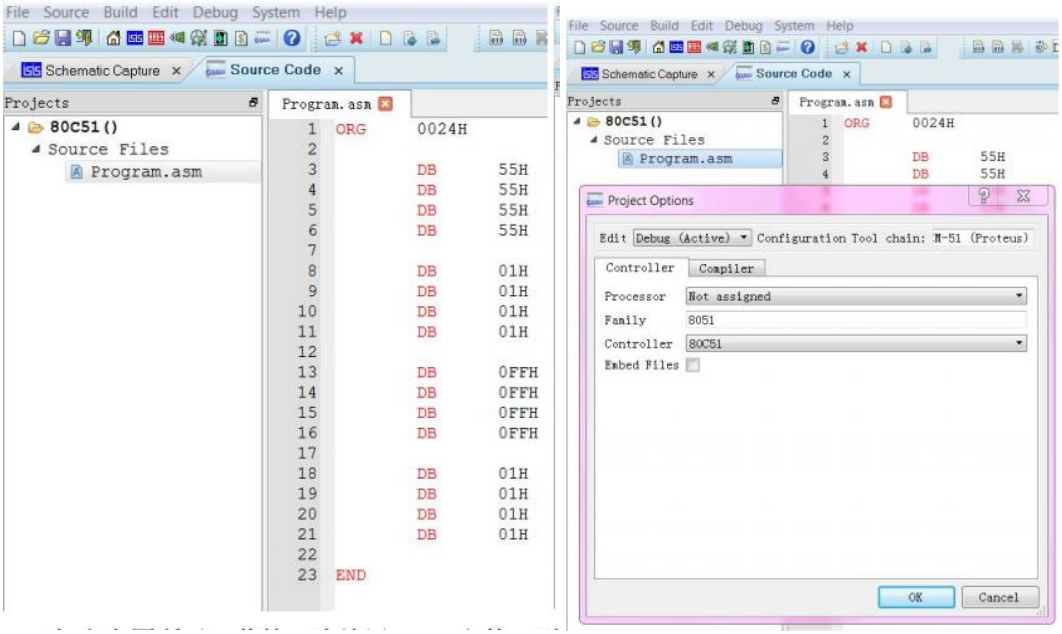


如果打开的标签页为空，则需要在菜单栏里选择 Source->Create Project，弹出的对话框如左下图所示。此处，在 Family 选项中选择“8051”，Controller 选项中选择“80C51”，则 Compiler 选项会自动选择“ASEM-51”；注意：下方 Crate Quick Start Files 选项不勾选！最后按下“确定”按钮。如左下图。



如右上图所示，标签页上会出现“*80C51()”的新 project 后，右键单击下方的“Source Files”，在弹出菜单中选择 Add files（或 Import Existing File）选项，在弹出的视窗中选择所需的 asm 文件，按“确定“就添加到当前的 project 中。

3）在 project 中已经添加的 asm 文件上双击，右侧会打开 asm 文本内容，如左下图所示。如果该 project 之前已经添加且编译过 asm 文件，则弹出的 Source Code 标签页右侧会直接显示 asm 文本内容，可以直接对 asm 文本内容进行修改和保存，如左下图所示。如果要更换 asm 文件，可以右键单击 asm 文件，菜单列表中选择“Remove file”删除当前 asm 文件，然后选择 Add files（或 Import Existing File）选项，在弹出的视窗中重新选择新的 asm 文件。

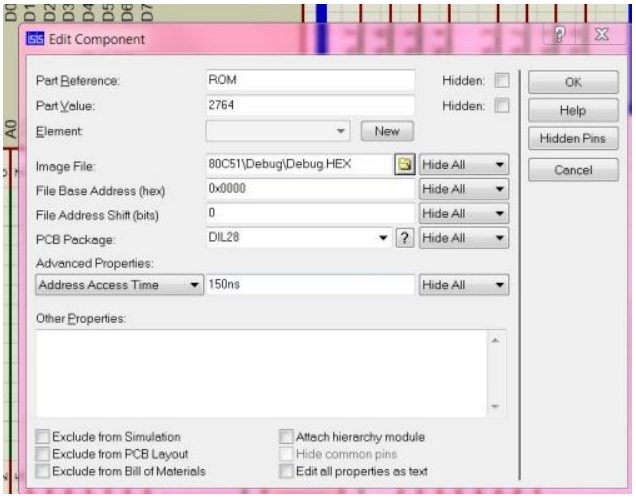


如右上图所示，若第一次编译 asm 文件，则须右键单击 asm 文件，在弹出菜单中点击 Project Settings，弹出框中 controller 选项选择“80C51”，注意：要去掉勾选 Embed files（或 Attach files ）!确保编译生成的 hex 文件在 project 当前目录下，最后点击“OK”

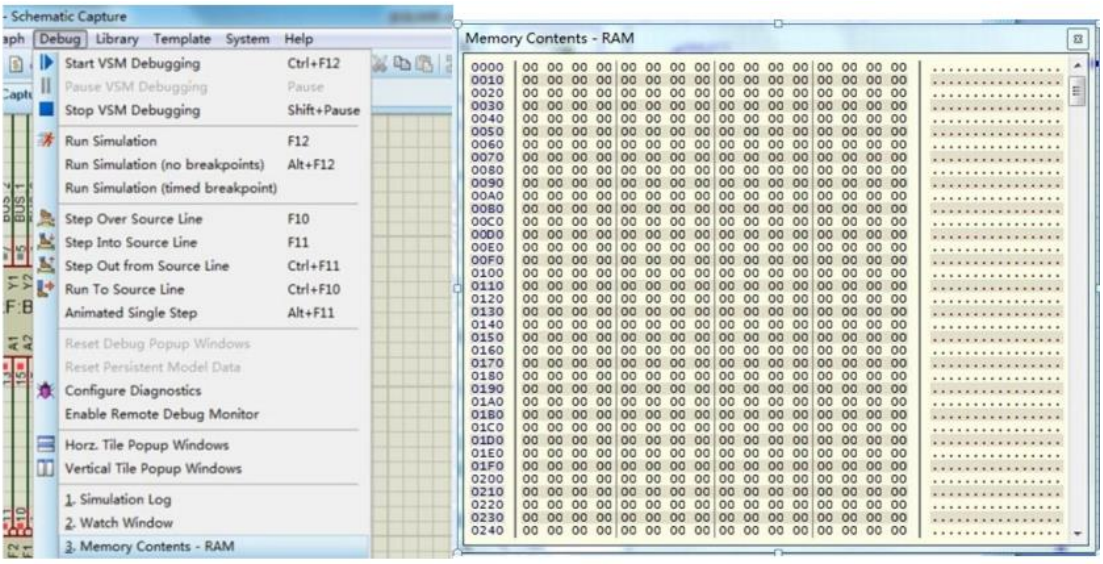
选项。然后，右键单击 asm 文件，在弹出框内点击 Build Project 执行编译。编译成功后，在下方的显示栏里会出现“Compiled successfully！”字样。此时，在项目当前文件夹的子文件夹 80C51 的文件夹 Debug 里面，就有编译后生成的 hex 二进制文件（请注意看文件的修改日期，确认是最近编译的文件）。该 hex 文件默认编译后的文件名为“Debug.hex”。

注：一个良好的编程习惯是每次编译生成 hex 文件后，就重命名为跟 asm 源程序一致的文件名，然后与同名 asm 源程序放在另外的固定目录下。否则下一次编译其他 asm 源程序的时候，会在相同路径生成重名的 Debug.hex，把以前编译的 hex 文件覆盖。

4）双击 ROM 芯片，弹出如下图的对话框：在 Image File 中选择所需烧写的 hex 文件，点击“OK“，ROM 加载成功，操作结束。 注：尽量避免直接加载\80C51\Debug 路径里的 Debug.hex 文件，因为每一次编译 asm 源程序，Debug.hex 都会自动刷新。但是，相同的文件名很容易混淆，难以确定加载的 hex 文件与哪个 asm 源文件对应。



5）启动 Proteus 仿真，在仿真过程下按“暂停”；点击菜单栏里的 Debug，在弹出的菜单 列表最下方有 Memory Contents-ROM/RAM，如左下图所示。项目里有多少个存储器，就有 多少个 Memory Contents 可供选择。点击所需查看存储器对应的 Memory Contents-选项，则 弹出如右下图所示的当前存储器内容：其中蓝色为该行的首地址（对应的是左边第一个黑色 数据 xxH），黑色为存储单元中的数据，按顺序从左到右，从上到下排列显示。可以通过调 整显示框的宽度来改变蓝色的行首地址显示，方便查看某一个固定的地址。值得注意的是， 右下图中显示的存储器地址是存储器地址线（例如本实验的 A0-A7）所定义的地址。而实际的地址空间不仅要看存储器的地址线，还要参考各个存储器片选信号的译码逻辑电路。



注：如果仿真出现以下报错，可能是项目路径里面原来加载 hex 文件的路径被取消，或是被移动到不同路径文件夹，或是文件名变了。特别是在项目里面有多个 ROM 需要加载的时候，要留意在移动或修改项目的时候不要破坏原有的 ROM 加载路径。

五、实验步骤

1）完成电路图，依照上述“ROM 批量导入数据”叙述的方法，加载 .asm 文件编译的 hex 二进制文件到 ROM 芯片 2764，并且查看 ROM 烧写的数据段是否正确。

2）启动仿真前，令 $\overline{ROM\_OE} = \overline{RAM\_OE} = \overline{RAM\_WE} = 1$；启动仿真后，令 $\overline{SW\_BUS} = 0$，手动拨码开关输入 024H 到地址总线 ABUS_[0..11]（绿色数码管显示）。

3)令地址锁存信号 ROM_CLK 上升沿跳变“0->1”，将地址总线上的 024H 打入地址锁存器 ROM_AR；令 $\overline{ROM\_OE} = 0$，使 ROM 存储器 2764 输出，在数据总线（红色数码管显示）上查看存储单元[024H]读出的内容。

4)手动拨码开关输入 F80H，令地址锁存信号 RAM_CLK 上升沿跳变“0->1”，向地址锁存器 RAM_AR 打入地址 F80H；令 $\overline{RAM\_WE} = 0$，使 RAM 存储器 6116 输入，把存储单元[024H]的内容写入存储单元[F80H]。再令 $\overline{RAM\_WE} = 1$，结束对 RAM 存储器的写入操作。

5）令 $\overline{ROM\_OE} = 1$（禁止 ROM 存储器 2764 输出）且 $\overline{RAM\_OE} = 0$，（允许 RAM 存储器 6116 输出），在数据总线上观察存储单元[F80H]写入内容是否正确。

6）按照上述操作，把 ROM 存储器单元[024H]、[028H]、[02CH]、[030H]的内容依次写入 RAM 存储器单元[F80H]、[F81H]、[F82H]、[F83H]，查看写入 RAM 的数据是否正确。

六、思考题

1、假设把 .asm 文件中的某个 ORG 语句改为“ORG 0224H”，请问该 ORG 定义的数据段还能被访问到么？如果不能，是数据批量导入 ROM 出错么？请修改 ROM 的地址片选电路，保证“ORG 0224H”所定义的数据段能被访问到。

2、为何 ROM 和 RAM 需要使用两个独立的 3-8 译码器？假设 RAM 的片选电路与 ROM 的片选电路共用一个 3-8 译码器，即 ROM 所在 3-8 译码器的最低 2 个端口给 ROM 使用，最 高 1 个端口给 RAM 使用。请给出 ROM 和 RAM 的地址空间范围。

3、假设 RAM 的地址空间范围改为 800H-8FFH，请问存储器地址片选电路如何修改？假设再把 ROM 的地址空间范围改为 600H-7FFH，请问存储器地址片选电路又如何修改？