第三章

Q3-1

1. 省略了 I/O 操作的复杂逻辑，易实现，耗费低；
2. 可以利用丰富的内存寻址模式实现灵活的 I/O 操作。

Q3-2

假设存储单元 ds1 处寄存器地址为 0x2000，代码如下

```
#define ds1 0x2000
while ( *ds1 == 0 ) ;
…
```

Q3-3

假设设备（dev1）中有两个寄存器 ds1 和 dd1，dev1 的地址为 0x1000，ds1 的偏移量（offset）为 0，则 ds1 地址为 0x1000，同理 dd1 的地址为 0x1004，代码如下

```
#define based_addr 0x1000
#define ds1 (based_addr + 0)
#define dd1 (based_addr + 4)
int data_dd1 ;
while ((peek (ds1) & 0x0001) == 0) ;

data_dd1 = peek (dd1) ;
```

Q3-4

假设存在一个设备 dev1，我们用 ARM 汇编语言实现对 dev1 的读写

```
dev1 EQU 0x1000
;;;peek()
LDR     r1, #dev1       ;
LDR     r0, [r1]        ;

;;;poke()
LDR     r2, #0x2        ;
STR     r2, [r1]        ;
```

Q3-19

a. 强制性未命中 compulsory miss：发生在单元第一次被访问时；
b. 容量未命中 capacity miss：工作集大于高速缓存容量；
c. 冲突未命中 conflict miss：两个地址映射到高速缓存的同一个单元。

Q3-20

$t_{av} = ht_{cache} + (1-h)t_{main}$　　$t_{av} = 10.25$ ns

Q3-21

$t_{av} = ht_{cache} + (1-h)t_{main}$　　$h = 98\%$

Q3-22

$t_{av} = h_1t_{L1} + h_2t_{L2} + (1-h_1-h_2)t_{main}$ 其中 $h_2=0.97 \times 0.1=0.097$　　$t_{av} = 5.295$ ns

Q3-23

将地址的最后一位作为组的索引（index）

存取 001 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	-	-	-	-
1	00	1111	-	-

存取 010 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	01	0000	-	-
1	00	1111	-	-

存取 011 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	01	0000	-	-
1	00	1111	01	0110

存取 100 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	01	0000	10	1000
1	00	1111	01	0110

存取 101 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	01	0000	10	1000
1	10	0001	01	0110

存取 111 后

组	块 0 标记	块 0 数据	块 1 标记	块 1 数据
0	01	0000	10	1000
1	10	0001	11	0100

Q3-24

假设程序在程序存储器中的地址分别为 0000、0001、0010、0011、0100、0101、0110、0111、1000、1001、1010、1011、1100（从上到下）

a.

每条指令执行 1 次，到 B loop，高速缓存的状态

块	标记	指令
00	11	B loop
01	10	MUL r4, r4, r6
10	10	ADD r2, r2, r4
11	10	ADD r0, r0, #1

循环执行完，高速缓存的状态

块	标记	指令
00	11	B loop
01	01	CMP r0, r1
10	01	BEG loopend
11	10	ADD r0, r0, #1

b.

每条指令执行 1 次，到 B loop，高速缓存的状态

块	标记	指令
000	1	LDR r6, [r5, r0]
001	1	MUL r4, r4, r6
010	1	ADD r2, r2, r4
011	1	ADD r0, r0, #1
100	1	B loop
101	0	CMP r0, r1
110	0	BEG loopend
111	0	LDR r4, [r3, r0]

循环执行完，高速缓存的状态

块	标记	指令
000	1	LDR r6, [r5, r0]
001	1	MUL r4, r4, r6
010	1	ADD r2, r2, r4
011	1	ADD r0, r0, #1
100	1	B loop
101	0	CMP r0, r1
110	0	BEG loopend
111	0	LDR r4, [r3, r0]

c.（采用 LRU 替换原则）

每条指令执行 1 次，到 B loop，高速缓存的状态

组	块 0 标记	块 0 指令	块 1 标记	块 1 指令
00	10	LDR r6, [r5, r0]	11	B loop
01	10	MUL r4, r4, r6	01	CMP r0, r1
10	10	ADD r2, r2, r4	01	BEG loopend
11	10	ADD r0, r0, #1	01	LDR r4, [r3, r0]

循环执行完，高速缓存的状态

组	块 0 标记	块 0 指令	块 1 标记	块 1 指令
00	10	LDR r6, [r5, r0]	11	B loop
01	10	MUL r4, r4, r6	01	CMP r0, r1
10	10	ADD r2, r2, r4	01	BEG loopend
11	10	ADD r0, r0, #1	01	LDR r4, [r3, r0]

Q3-27

取指、译码、执行

Q3-29

指令延迟：指令从开始执行到结束的时间

指令吞吐量：单位时间执行的指令数

Q3-31

电压降、切换、泄漏

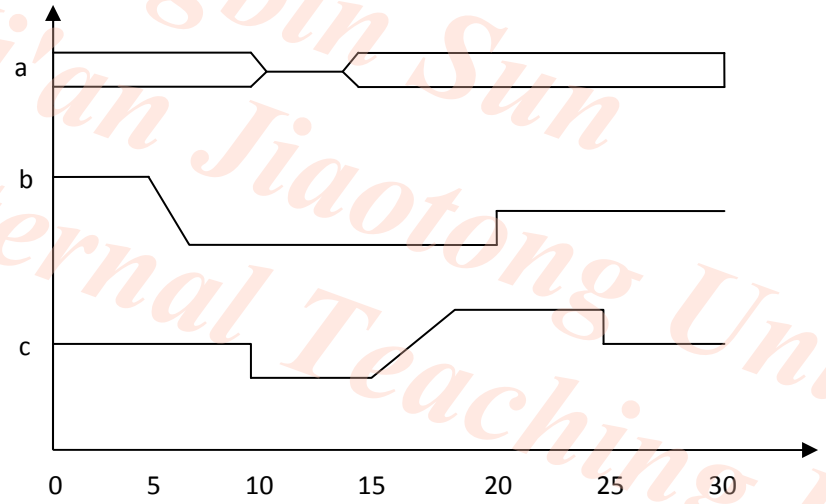
Q3-32

a．节电模式

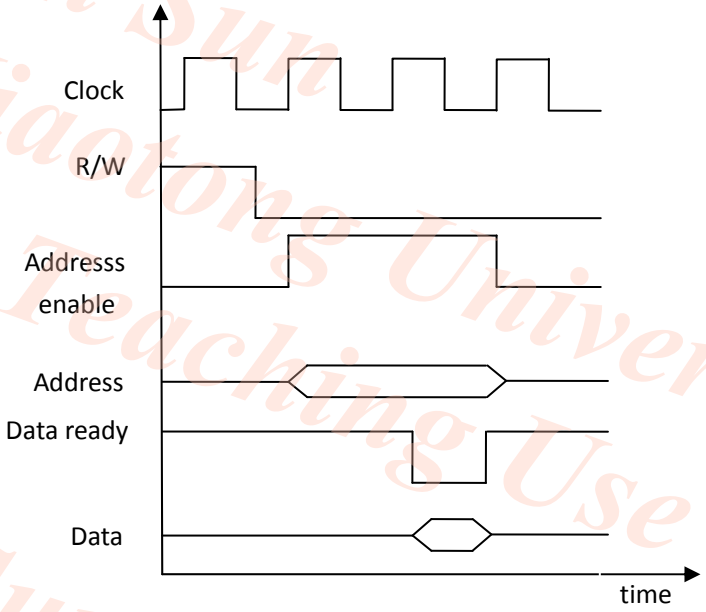
b．当指令运行时，CPU 会关掉部分不需要运行的指令，从而减少功耗

第四章

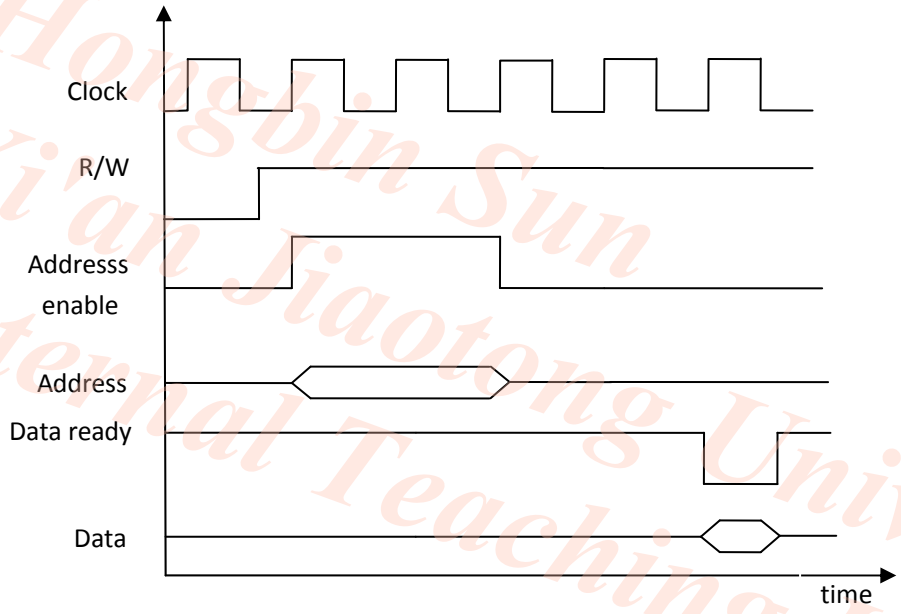
Q4-2



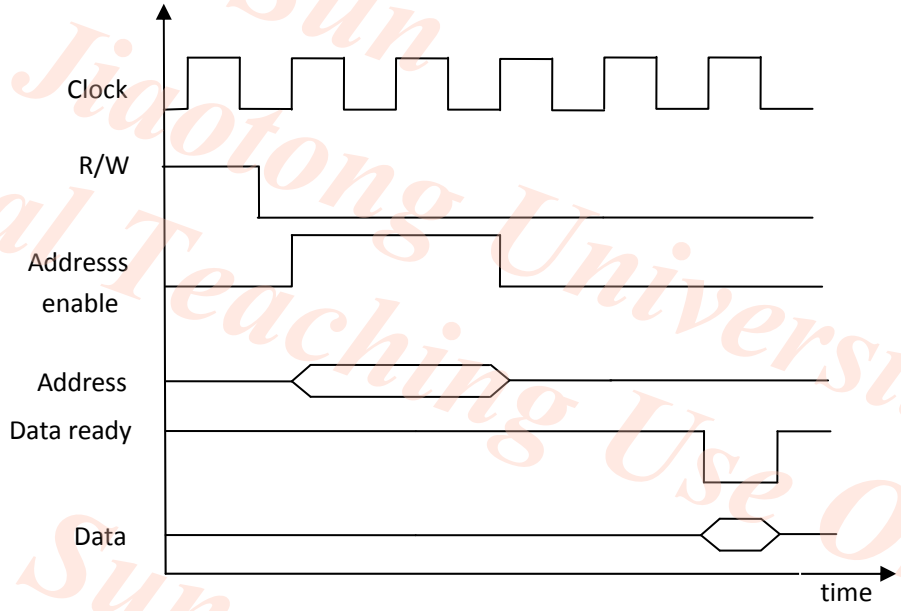
Q4-3



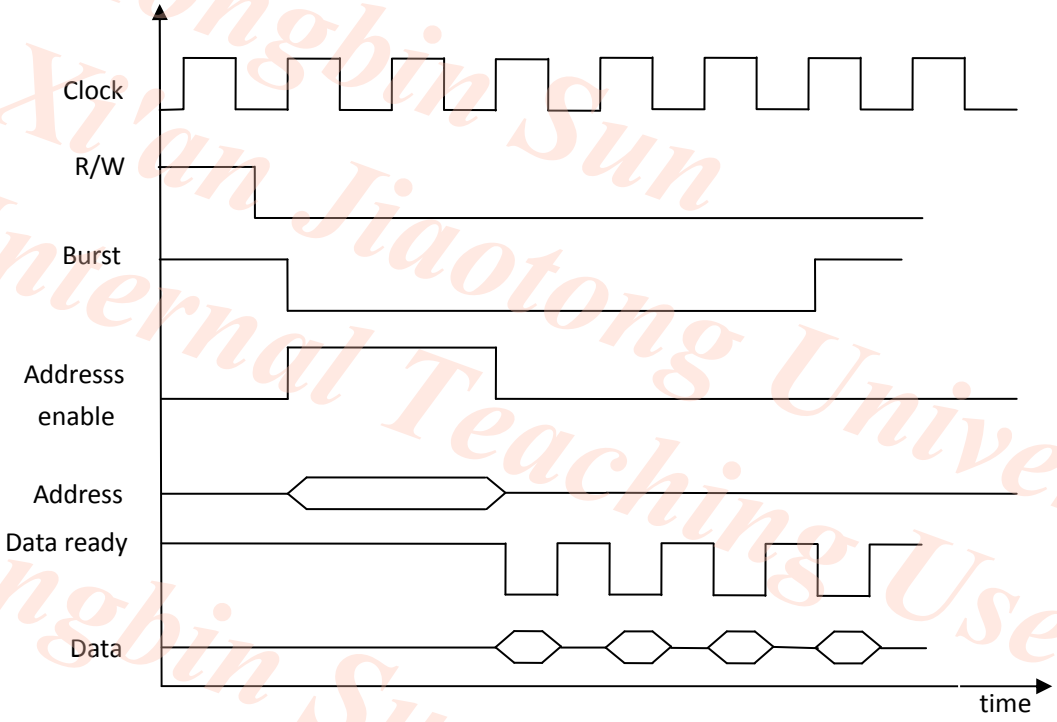
Q4-4



Q4-5



Q4-6



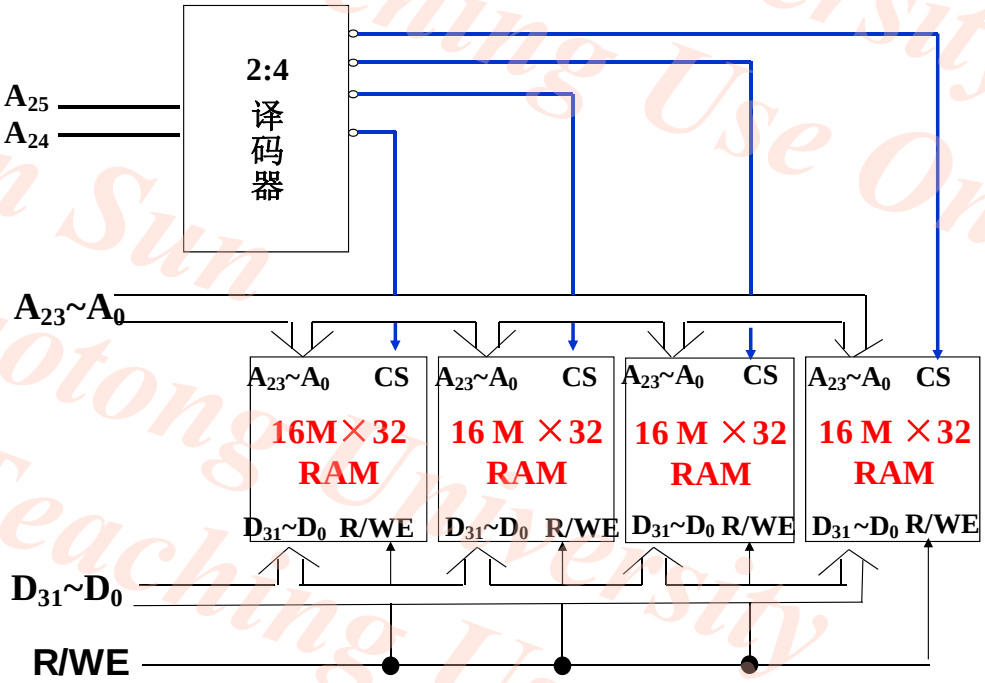
Q4-18

A：地址总线

D：数据总线

CS：片选信号

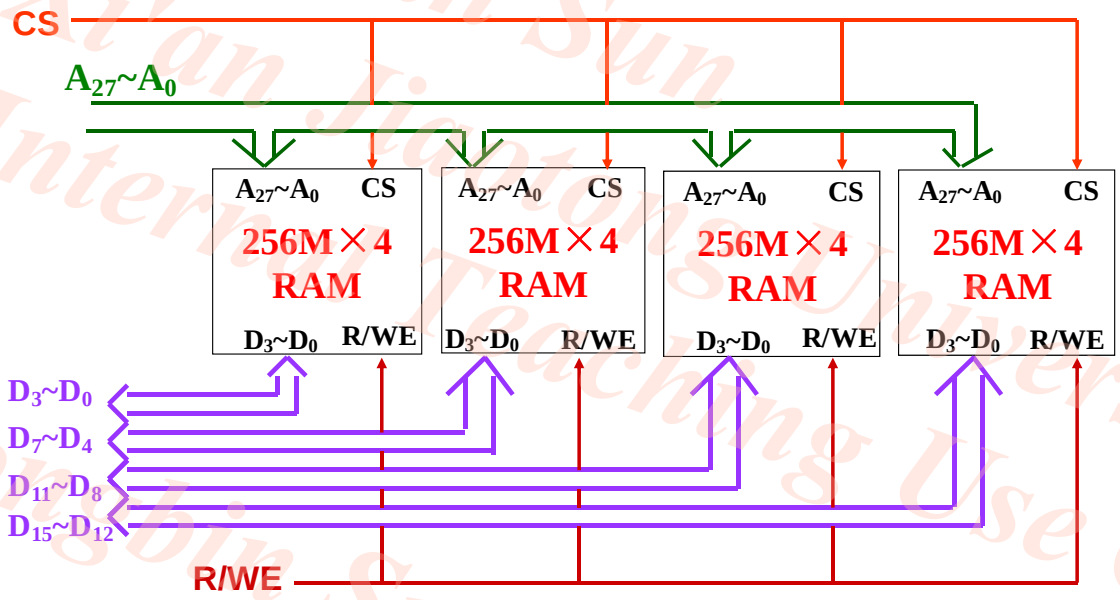
R/WE：读写信号 （CS、WE 低电平有效）



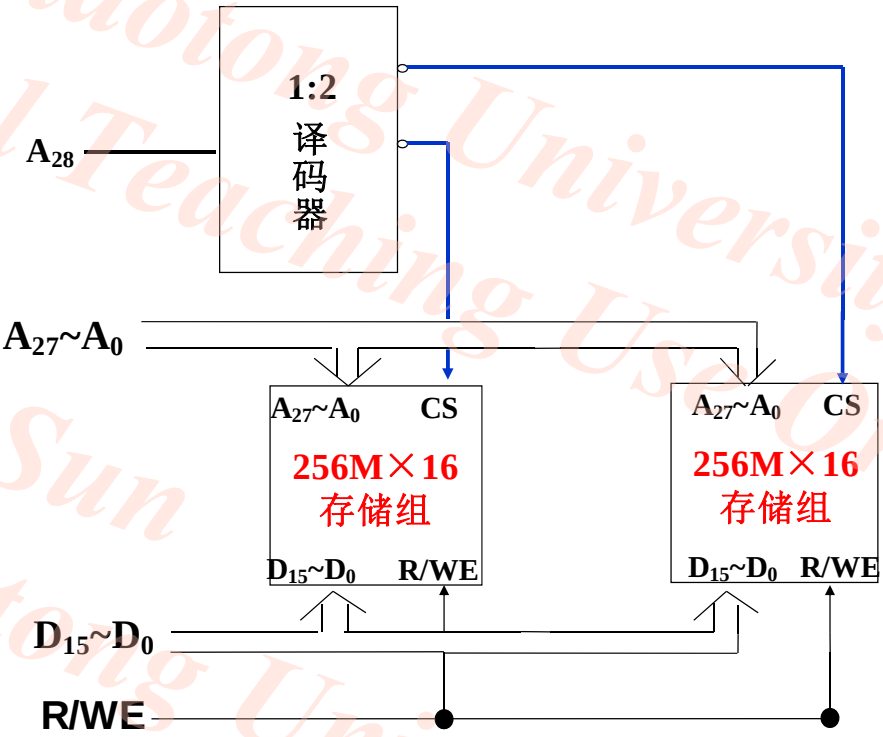
Q4-19

分两步走：位扩展和字扩展，其中 CS、WE 都是低电平有效

1. 位扩展



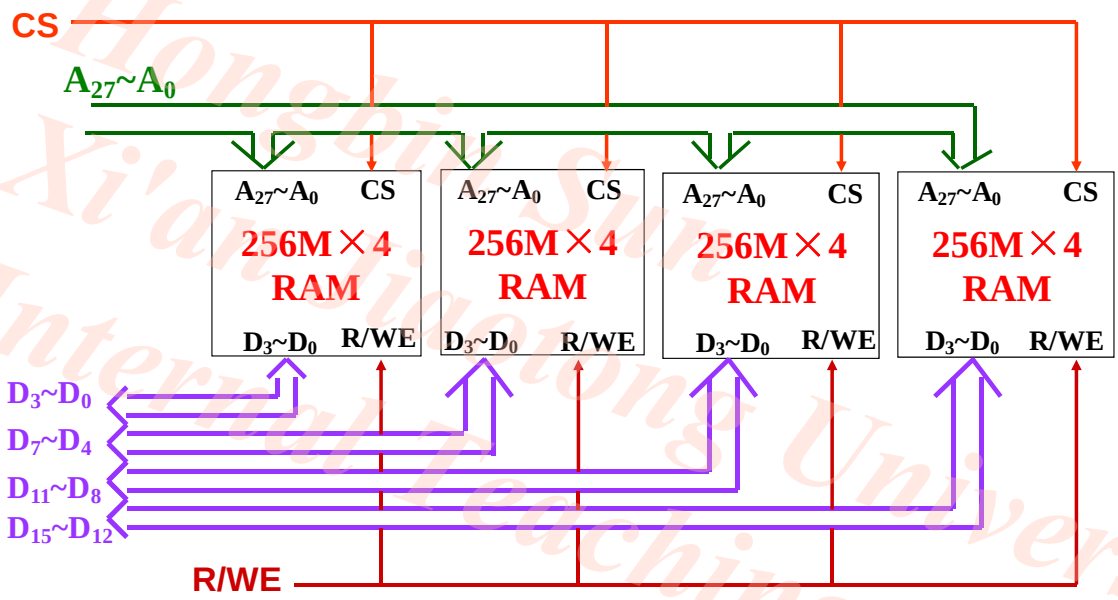
2. 字扩展



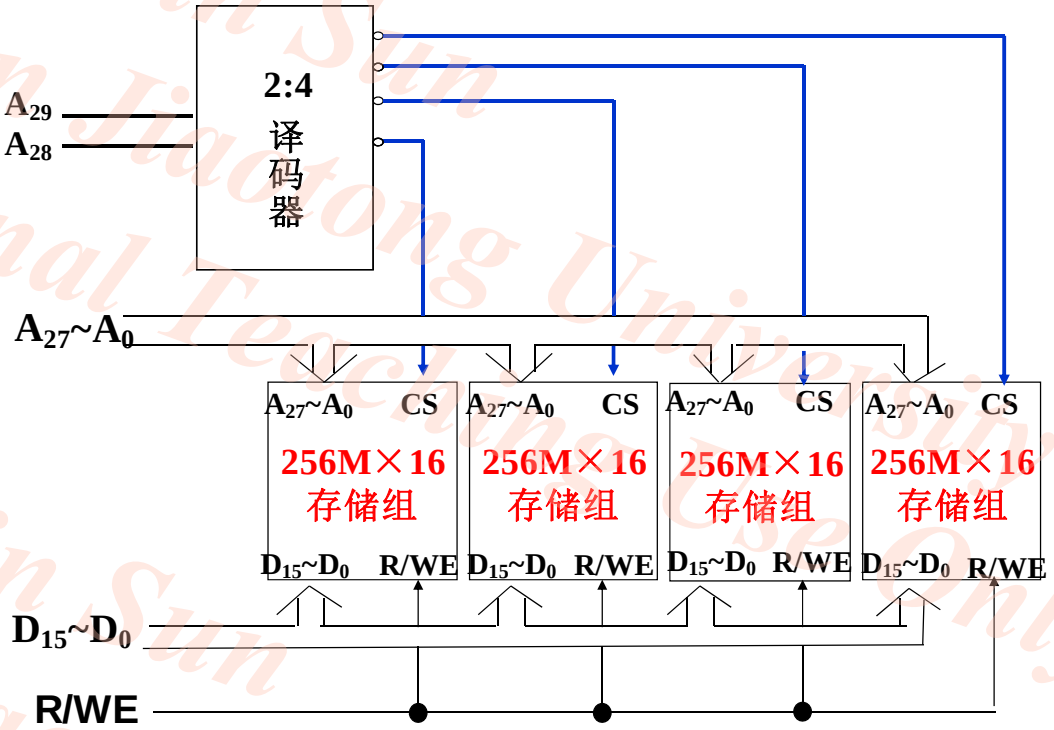
Q4-20

同上题一样，先位扩展，在字扩展

1. 位扩展，成 256M×16 存储组



2. 字扩展，成 1G x 16 存储器



Q4-25

Tcpu <= 1/44.1khz = 2.3 x 10^{-5} s

20MHZ = 5 x 10^{-8} s　执行指令条数 n = 2.3 x 10^{-5} / 5 x 10^{-8} - 100 =360

Q4-26

如果下次发生的中断优先级低于或等于前次，则正在执行的中断服务子程序继续执行；若下次发生的中断优先级高于前次，则正在执行的中断服务子程序被打断，中断服务程序优先处理后发生中断之后，然后再处理前次中断。