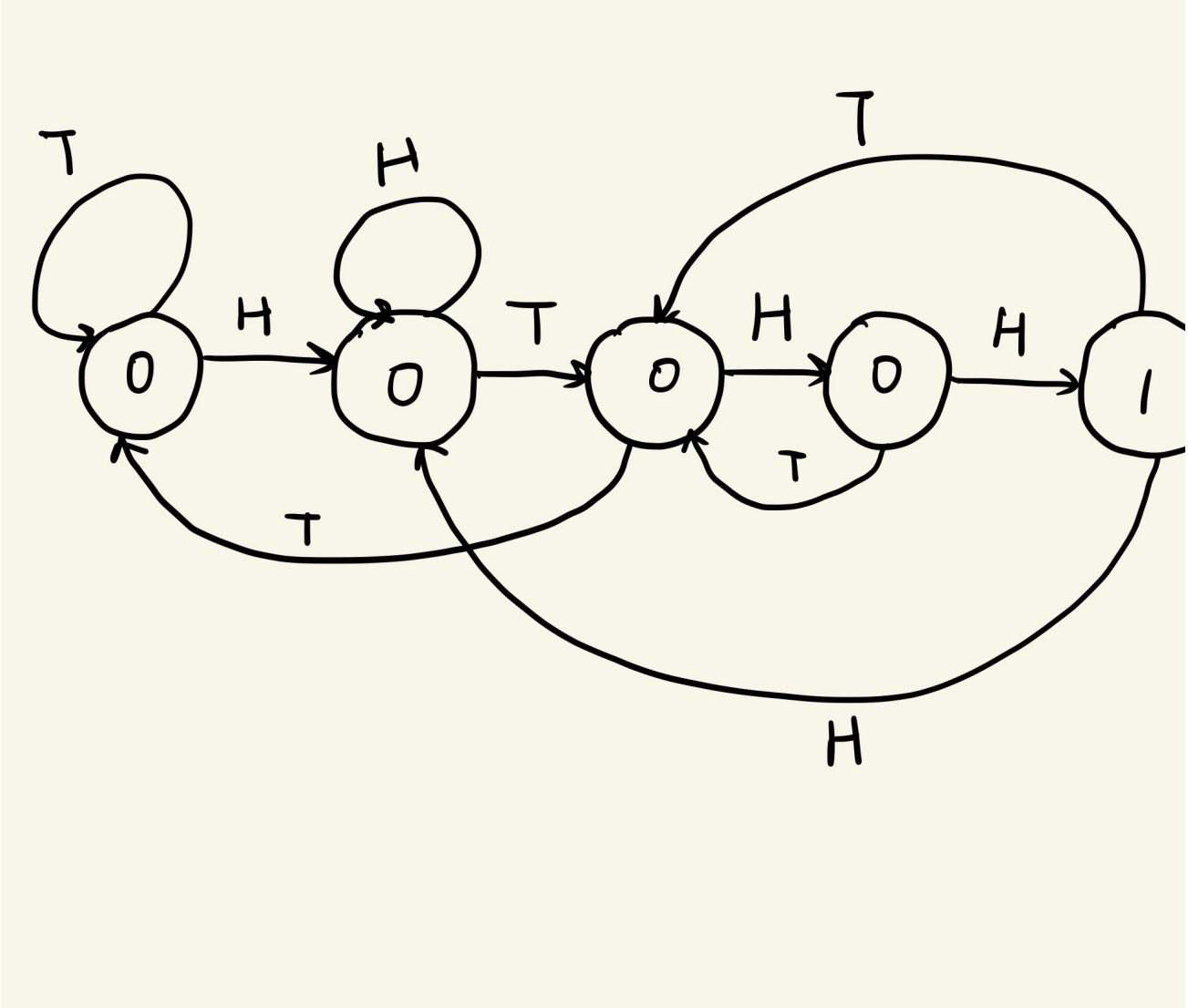
HW03

一：

1. 补充状态图如下：



2. 共五个状态，且无法进一步化简，所以最少需要三个变量来表示状态

二：

2^7 * 16 = 2^11 = 2048 (byte)

三：

1. A[1:0]为地址线，应为 10
   WE 控制 “wtite” 操作时值应为 1
2. 2^9 < 800 <2^10 故需要 10 根 address line

而改变地址空间后，寻址能力不变，仍为 3

3. 2^10 – 800 = 1024 – 800 = 224

四：

1. 2^2 = 4
2. 16-bit
3. 2^2 * 16 = 64 (bit)

WE	A[1:0]	Di[15:0]	D[15:0]	R/W
0	01	xFADE	x4567	R
1	10	xDEAD	xDEAD	W
0	00	xBEEF	x0123	R
1	11	xFEED	xFEED	W

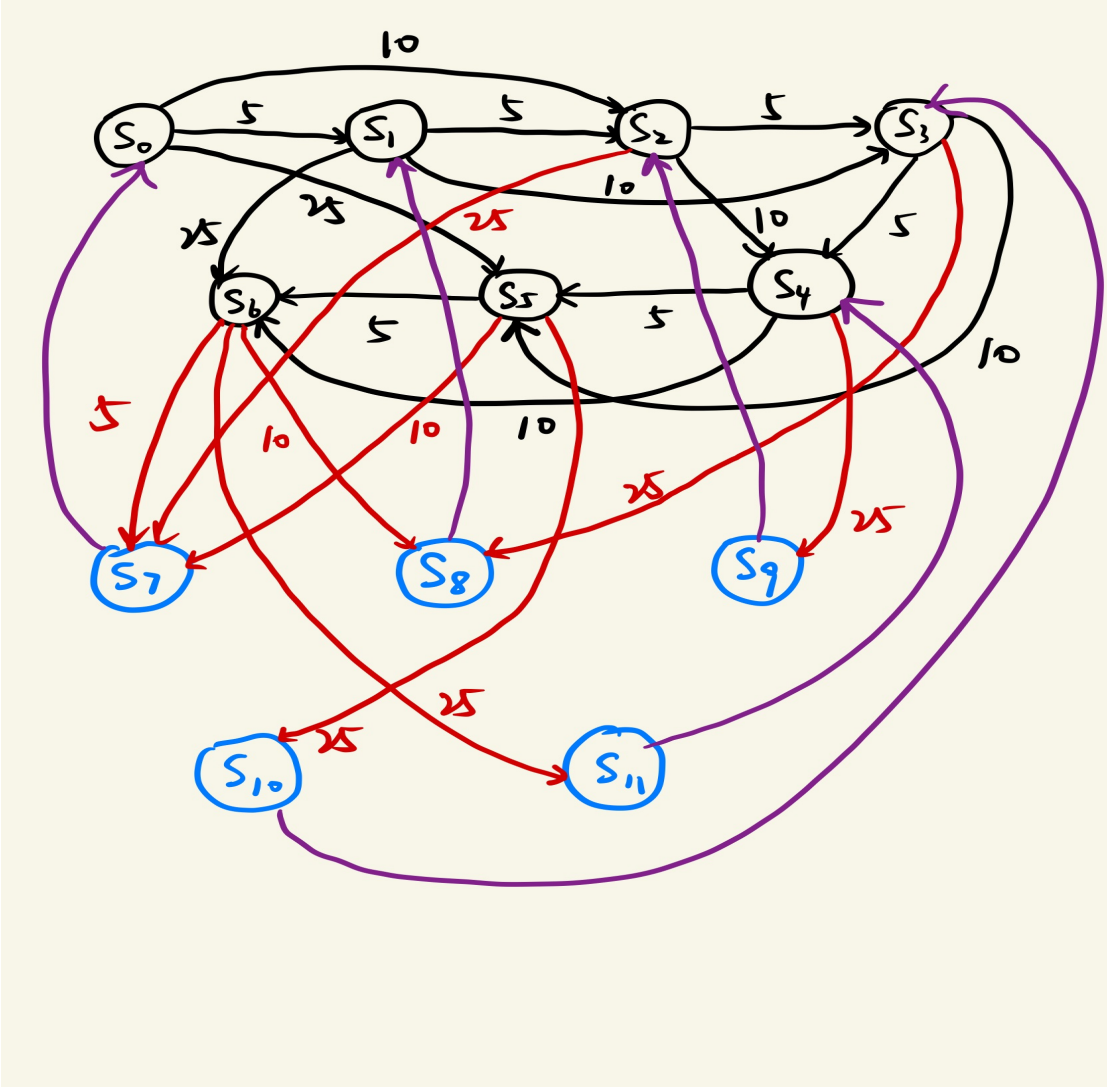
五：

nickel —— 5 cents

dime —— 10 cents

quarter —— 25 cents

作出状态图如下：



其中黑色部分代表投入硬币总数<35 的 7 个状态，蓝色部分代表硬币总数>=35 的 5 个状态，红线代表出苏打水前的最后一次投币，紫线代表返回状态

六：

x62BE 为 0110　001　010　111110

即将寄存器 R2 的内容加（- 2），求得内存地址，然后将该地址指向的内存单元内容读入，存放在寄存器 R1 中

	PC	IR	MAR	MDR	R0	R1	R2	R3	R4	R5	R6	R7
fetch	x3004	x62BE	x3003	x62BE	x3000	x3000	x3002	x3000	x3000	x3000	x3000	x3000
decode	x3004	x62BE	x3003	x62BE	x3000	x3000	x3002	x3000	x3000	x3000	x3000	x3000

EA	x3004	x62BE	x3003	x62BE	x3000	x3000	x3002	x3000	x3000	x3000	x3000	x3000
FO	x3004	x62BE	x3000	x62BF	x3000	x3000	x3002	x3000	x3000	x3000	x3000	x3000
execute	x3004	x62BE	x3006	x62BF	x3000	x3000	x3002	x3000	x3000	x3000	x3000	x3000
SR	x3004	x62BE	x3006	x62BF	x3000	x62BF	x3002	x3000	x3000	x3000	x3000	x3000

七：

1. 2^10 < 1511 < 2^11　故最少需要 11 位
2. 2^5 < 40 < 2^6 故最少需要 6 位
3. 32 – (11 + 3*6) = 3 所以最多有 3 位

八：

```
0101 000 000 1 00000
1001 100 001 111111      //将寄存器 R1 中的值按位取反放入寄存器 4 中
0001 100 100 1 00001     //将寄存器 R4 中的值加 1
0001 100 011 000 100     //R4<–R4+R3
0000 010 00000010        //若相等（相减为零）则跳两步置 1
0001 000 000 1 00010     //若不等则置 2
1111 0000 0010 0101
0001 000 000 1 00001
1111 0000 0010 0101
```

九：

具体分析如下：

```
0101 100 100 1 00000     //将寄存器 R4 清零
1001 000 001 111111      //将寄存器 1 中的值按位取反放入寄存器 0 中
0001 000 000 1 00001     //将寄存器 0 中的值加 1
0001 000 000 000 010     //将寄存器 2 与寄存器 0 的值相加，结果放入寄存器 0 中
0000 100 000000001       //若上一步的运算结果为负数，则直接终止程序
0001 100 100 1 00001     //R4<–R4+1
1111 0000 0010 0101      //终止程序
```

其中 2-3 步相当于将 R1 中的值的相反数存入了 R0 中，故第四部得到的结果实际是 R2 减去 R0，故该程序的作用是：

若 R2 中的数据不小于 R1 中的数据，则将 R4 计 1；否则 R4 计 0.