

## 一、填空

1. (1)电子管; (2)晶体管; (3)半导体; (4)微处理器;
2. (5)光刻; (6)栅极; (7)摩尔;
3. (8)存储程序; (9)程序控制; (10)存储器; (11) 控制器;
4. (12) 逻辑门 (13) 微代码 (14) 指令集架构 (15) 汇编语言
5. (16) 发现错误 (17) 纠正错误 (18) 冗余 (19) 码距 (20) 4 (21) 两 (22) 一 (23) 而 (24) 唯一性
6. (25) 行波 (26) 先行 (27) 生成 (28) 传递
7. (29) 信息量 (30) 周期 (31) 位宽
8. (32) 易失 (33) 负载 (34) 双稳态 (35) 地址 (36) 读写控制
9. (37) 电容 (38) 预充 (39) 选通 (40) 放大 (41) 恢复 (42) 放大检测
10. (43) 开关 (44) 熔丝 (45) 1 (46) 浮栅型 (47) 紫外线 (48) 某一 (49) 闪存
11. (50) 多字存储器 (51) 低位交叉 (52) 流水线
12. (53) 调入 (54) 一致性 (55) 写回 (56) LRU
13. (57) 存储管理控制部件 MMU (58) cache (59) 全相联 (60) 页表项。

## 二、简答题

1. (1) 根据操作数和运算结果的符号位是否一致进行检测。(符号位检测法)

正正得负、负负得正时溢出。

设  $X_f$ ,  $Y_f$  为操作数的符号位,  $S_f$  为运算结果的符号位,  $V=0$ , 没有溢出;  $V=1$ , 有溢出。

溢出标志  $V$  检测公式:  $V = X_f Y_f / S_f + /X_f / Y_f S_f$

如果是减法, 则溢出标志  $V$  的检测公式为:  $V = X_f / Y_f / S_f + /X_f Y_f S_f$

$V=1$  表示溢出,  $V=0$  表示无溢出。

- (2) 根据运算过程中最高数据位的进位与符号位的进位是否一致进行检测。(进位标志异或法)

$C_f$ : 符号位产生的进位信号;  $C_d$ : 最高有效位产生的进位信号。

溢出标志  $V$  检测公式:  $V = C_f \oplus C_d$

$V=1$  表示溢出,  $V=0$  表示无溢出。

(3) 利用变形补码的符号位进行检测 (双符号位检测法)

如果运算结果的符号位为 00 或 11, 则没有溢出; 如果运算结果的符号位为 01 或 10, 则产生溢出。Sf1、Sf2 分别表示运算结果的第 1 个符号位和第 2 个符号位。

溢出标志 V 检测公式:  $V = S_{f1} \oplus S_{f2}$

V=1 表示溢出, V=0 表示无溢出。

2. (1)

主存容量为 1MB=2<sup>20</sup>B, Cache 块大小为 16B=2<sup>4</sup>B, 故块内偏移位为 4 位。

Cache 总容量为 16KB=2<sup>14</sup>B, 包含 16KB/16B=1024 个块, 因此行索引位为 10 位。

主存地址剩余部分为标识位 (区地址), 即 20-10-4=6 位。

**标识位 (区地址):** 前 6 位 101010,

**行索引:** 中间 10 位 1111001101,

**块内偏移:** 末 4 位 1110。

因此, 该主存单元映射到 Cache 的 **第 1111001101 行**, 对应块内第 1110 字节 (十进制 14)

(2) 四路组相联即每组包含 4 个块, Cache 总块数为 16KB/16B=1024 块, 共分为 1024/4=256 组, 故组索引位为 8 位。

主存地址格式为: **标记位 (区地址) + 组索引 + 块内偏移**, 其中块内偏移仍为 4 位, 组索引 8 位, 剩余 20-8-4=8 位为标记位。

对地址 1010 1011 1100 1101 1110 分解:

**标记位:** 前 8 位 10101011,

**组索引:** 中间 8 位 11001101,

**块内偏移:** 末 4 位 1110。

该地址对应 Cache 的 **第 11001101 组**, 可映射到该组 4 个块中的任意一块, 块内位置为第 1110 字节。

3. (1) 使部分积的相加运算始终在固定位置上进行, 同时将 2n 位长度的加法器变成 n 位长度, 减少硬件开销。

(2) 由  $y_n$  决定是否加被乘数  $|x|$ 。  $y_n=1$  时, MUX 输出  $|x|$ , 与部分和相加;  $y_n=0$  时, MUX 输出 0。与原码一位乘的规则一致。

(3)  $R_0$  初始存放部分积 0, 结束时存放乘积的高  $n$  位。  $R_1$  初始存放乘数  $y$ , 结束时存放乘积的低  $n$  位。

(4) 设被乘数为  $X$ , 乘数为  $Y$ , 部分积为  $P$ , 则原码一位乘的递归公式为:

$P_0=0; P_{i+1}=(P_i+Y_i \times X) \gg 1, i=0,1,\dots,n-1$  其中  $Y_i$  为乘数  $Y$  的第  $i$  位,  $\gg 1$  表示右移一位。

(5) ①  $C_{in}$  是加法器的进位输入, 用于在补码一位乘运算中处理符号位的进位。在补码一位乘中, 需要根据乘数的相邻两位  $y_n$  和  $y_{n+1}$  来确定加法操作。

$C_{in}$  与  $y_n y_{n+1}$  的关系是:  $C_{in}=1$  时, 可以支持  $[-x]_{补}$  的运算。即  $[x]_{反}+1$ , 而其又与  $y_n/y_{n+1}$  存在下述关系:

$C_{in}=0$  时,  $y_n y_{n+1} = 00, 01, 11$

$C_{in}=1$  时,  $y_n y_{n+1} = 10$

②在原码一位乘中, 符号位是单独处理的, 运算过程中只涉及绝对值的乘法, 所以进位输出  $C_{out}$  参与移位运算, 它表示了部分积在加法运算中的进位情况, 通过移位可以正确地将进位传递到相应的位上, 以得到正确的乘积结果;

在补码一位乘中, 符号位和数值位一起参与运算, 加法器的进位输出  $C_{out}$  不参与移位运算, 因为补码乘法采用了特定的算法, 通过对乘数的相邻两位进行判断来确定加法操作, 而不是像原码乘法那样简单地根据进位来进行移位。  $C_{out}$  仅用于更新  $C_{in}$ , 以确保在进行下一次加法运算时能够正确处理进位。如果让  $C_{out}$  参与移位, 可能会导致符号位的错误, 因为补码的符号位是有特殊含义的, 不能简单地像原码那样进行移位操作。

4. (1) TLB 命中: 表示页表项 PTE 在 TLB 中, 且页表项 PTE 中有效位为 1。 cache 命中: 根据 PA 要访问的数据在 cache 中。 页命中: 根据页表项地址 PTEA 访问 cache 或主存中的页表, 返回的页表项 PTE 中有效位为 1

(2) 当 TLB 缺失, 且从 cache 中的页表中找不到页表项, 则需要访问主存的页表。

(3) 存在。此时, TLB 缺失; 将访问 cache 或主存中的页表, 如果返回的页表项 PTE 中的有效位为 1, 则表示页命中; 页命中后, 将根据 PA 访问 cache, 此时数据在 cache 中, 表示 cache 命中。

(4) 缺页处理需要访问磁盘, 而 cache 缺失只要访问主存; 因此, cache 缺失带来的开销小, 而处理缺页的开销大。

5.电路如下所示:

