HW2

Q1

Q1.1

汇编代码	对应函数
foo1	`int choice3(int x)`
foo2	`int choice5(int x)`
foo3	`int choice6(int x)`

Q1.2

空格	值
空格(1)	`%esi`
空格(2)	`%eax`
空格(3)	`%rdi`
空格(4)	`je`

Q1.3

case	执行语句
`default`	`w=2;break;`
`1`	`w=y*z;break;`
`2`	`w=y/z+z;break;`
`3`	`w+=z;break;`
`5` or `6`	`w-=z;break;`

Q2

Q2.1

表达式	T/F	解释
`x==(int)(float)x`	F	`x=3`
`(d+f)-d==f`	F	若当 `d+f` 超过 `double` 数据类型的数值上限, 将发生上溢
`if(x <= 0) then  -x >= 0`	F	若 `x==INT_MIN` , 则 `-x` 将仍是 `INT_MIN` (按位取反再加1)
`x >> 4 == x / 16`	F	如果 `x<0` , `x >> 4` 没有考虑偏置量
`(x|-x) >> 31 == -1`	F	`x==0` 显然不成立

Q2.2

`jg` 指令是用于带符号整数比较的.

首先可以将大于条件看成大于等于并且不等于, 这样 `~(SF^OF)` 就代表大于等于, `~ZF` 代表不等于.

由于 `ZF` 是零标志, 自然可以代表等于.

而 `SF` 是符号标志, 代表结果的符号, `OF` 是溢出标志, 代表结果是否溢出; 因此大于等于应该满足两数做差的结果是正数不溢出或者负数但溢出, 这样的逻辑恰好就可以用异或的逻辑表示, 然后再用非的逻辑将比较符号方向调整过来.

Q2.3

反汇编代码	对应的指示宏	理由
(1)	`[[unlikely]]`	因为在 `1080` 处的指令是 `je` , 意味着 `a==2` 才需要跳转, 对于流水线来说不跳转效率更高, 那么对应的就是 `a==2` 可能性更小
(2)	`[[likely]]`	与上面的逻辑正相反

因为这相当于给编译器提供了额外的信息, 编译器可以根据每种情况的概率大小来调整跳转的顺序, 从而提高程序的效率.