# Процесорот на Кибид

Кибид инжињерот работи на нов тип на компјутерски процесор.

Процесорот има пристап до $m$ различни $b$-битни мемориски ќелии (каде $m = 100$ и $b = 2000$), кои се нарекуваат **регистри**, и се означени од $0$ до $m - 1$. Ќе ги означуваме регистрите со $r[0], r[1], \ldots, r[m-1]$. Секој регистар е низа од $b$ битови, нумерирани од $0$ (најдесниот бит) до $b - 1$ (најлевиот бит). За секој $i$ $(0 \leq i \leq m - 1)$ и секој $j$ $(0 \leq j \leq b - 1)$, го означуваме $j$-тиот бит од регистарот $i$ со $r[i][j]$.

За било која секвенца од битови $d_0, d_1, \ldots, d_{l-1}$ (со произволна должина $l$), **целобројната вредност** на секвенцата е еднаква на $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$. Ќе кажеме дека **целобројната вредност зачувана во регистар** $i$ е целобројната вредност на секвенцата од неговите битови, т.е., зачуваната вредност е $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

Процесорот има $9$ видови на **инструкции** кои можете да ги користите за да ги модифицирате битовите во регистрите. Секоја инструкција извршува операции над еден или повеќе регистри и ја зачувува вредноста во некој од регистрите. Понатака во текстот, ќе користиме $x := y$ за да означиме операција на променување на вредноста на $x$ така да добива нова вредност $y$. Операциите изведени од секој вид инструкција се дадени подолу.

- $move(t, y)$: Копирај ја низата битови од регистар $y$ во регистар $t$. За секој $j$ $(0 \leq j \leq b - 1)$, постави $r[t][j] := r[y][j]$.
- $store(t, v)$: Постави го регистарот $t$ да биде еднаков на $v$, каде $v$ е низа од $b$ битови. За секој $j$ $(0 \leq j \leq b - 1)$, постави $r[t][j] := v[j]$.
- $and(t, x, y)$: Направи AND бит по бит (анг. bitwise-AND, пример $1100 bitwise-AND 0101 = 0100$) на регистрите $x$ и $y$, и зачувајте го резултатот во регистарот $t$. За секој $j$ $(0 \leq j \leq b - 1)$, постави $r[t][j] := 1$ ако **и двата од** $r[x][j]$ и $r[y][j]$ се $1$, а во спротивен случај постави $r[t][j] := 0$.
- $or(t, x, y)$: Направи OR бит по бит (анг. bitwise-OR, пример $1100 bitwise-OR 0101 = 1101$) на регистрите $x$ и $y$, и зачувајте го резултатот во регистарот $t$. За секој $j$ $(0 \leq j \leq b - 1)$, постави $r[t][j] := 1$ ако **барем еден** од $r[x][j]$ и $r[y][j]$ се $1$, а во спротивен случај постави $r[t][j] := 0$.
- $xor(t, x, y)$: Направи XOR бит по бит (анг. bitwise-XOR, пример $1100 bitwise-XOR 0101 = 1001$) на регистрите $x$ и $y$, и зачувајте го резултатот во регистарот $t$. За секој $j$ $(0 \leq j \leq b - 1)$, постави $r[t][j] := 1$ ако **точно еден** од $r[x][j]$ и $r[y][j]$ е $1$, а во спротивен случај постави $r[t][j] := 0$.

- $not(t, x)$: Take the bitwise-NOT of register $x$, и зачувајте го резултатот во регистарот $t$. За секој $j$ $(0 \leq j \leq b-1)$, постави $r[t][j] := 1 - r[x][j]$.

- $left(t, x, p)$: Помести ги сите битови на регистарот $x$ на лево за $p$, и зачувајте го резултатот во регистарот $t$. Резултатот од поместување на битовите од регистарот $x$ на лево за $p$ е низа $v$ која се состои од $b$ битови. За секој $j$ $(0 \leq j \leq b-1)$, $v[j] = r[x][j-p]$ ако $j \geq p$, и $v[j] = 0$ инаку. За секој $j$ $(0 \leq j \leq b-1)$, постави $r[t][j] := v[j]$.

- $right(t, x, p)$: Помести ги сите битови на регистарот $x$ на десно за $p$, и зачувајте го резултатот во регистарот $t$. Резултатот од поместување на битовите од регистарот $x$ на десно за $p$ е низа $v$ која се состои од $b$ битови. За секој $j$ $(0 \leq j \leq b-1)$, $v[j] = r[x][j+p]$ ако $j \leq b-1-p$, и $v[j] = 0$ инаку. За секој $j$ $(0 \leq j \leq b-1)$, постави $r[t][j] := v[j]$.

- $add(t, x, y)$: Собери ги целобројните вредности зачувани во регистрите $x$ и $y$, и зачувајте го резултатот во регистарот $t$. Собирањето се врши модуло $2^b$. Формално, нека $X$ целобројната вредност зачувана во регистарот $x$, и $Y$ нека е целобројната вредност зачувана во регистарот $y$ пред операцијата. Нека $T$ е целобројната вредност зачувана во регистарот $t$ после операцијата. Ако $X + Y < 2^b$, Поставит ги битовите на $t$, така да $T = X + Y$. Инаку, постави ги битовите на $t$, така да $T = X + Y - 2^b$.

Кибид би сакал да решите два вида на задачи користејќи го новиот процесор. Типот на задачата е означена со бројот $s$. За двата вида задачи, вие треба да доставите **програма**, која е секвенца од иснтрукциите наведени погоре.

**Влезот** од програмата се состои од $n$ цели броеви $a[0], a[1], \ldots, a[n-1]$, секој од кој се состои од $k$ битови, т.е., $a[i] < 2^k$ ( $0 \leq i \leq n-1$). Пред програмата да биде извршена, сите од броевите на влез се зачувани секвенцијално во регистарот $0$, така да зза секој $i$ $(0 \leq i \leq n-1)$ целобројната вредност на секвенцата од $k$ битови $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ е еднаква на $a[i]$. Забележете дека $n \cdot k \leq b$. Сите други битови во регистарот $0$ (т.е., тие со индекси помеѓу $n \cdot k$ и $b-1$, вклучително) и сите битови во сите други регистри се иницијализирани на $0$.

Извршувањето на програма се состои од извршување на нејзините инструкции по редослед. Откако последната инструкција е извршена, **излезот** на програмата се определува во однос на финалната вредност на битовите во регистарот $0$. Специфично, излезот е секвенцата од $n$ цели броеви $c[0], c[1], \ldots, c[n-1]$, каде за секој $i$ ( $0 \leq i \leq n-1$), $c[i]$ е целобројната вредност на секвенцата која се состои од битовите $i \cdot k$ to $(i+1) \cdot k - 1$ од регистарот $0$. Забележете дека после извршувањето на програмата преостанатите битови од регистарот $0$ (со индекси поголеми или еднакви на $n \cdot k$) и сите битови на сите останати регистри може да се произволни.

- Првата задача $(s = 0)$ е да го најдете најмалиот цел број од влезните цели броеви $a[0], a[1], \ldots, a[n-1]$. Специфично, $c[0]$ мора да е минимум од $a[0], a[1], \ldots, a[n-1]$. Вредностите на $c[1], c[2], \ldots, c[n-1]$ може да се произволни.

- Втората задача $(s = 1)$ е да ги подредите влезните цели броеви $a[0], a[1], \ldots, a[n-1]$ во неопаѓачки редослед. Специфично, за секој $i$ ( $0 \leq i \leq n-1$), $c[i]$ треба да е еднаков на $1+i$-тиот најмал цел број од $a[0], a[1], \ldots, a[n-1]$ (т.е., $c[0]$ е најмалиот цел број од влезните цели броеви).

Дадете му на Кибид програми, кои се состојат од најмногу $q$ инструкции секоја, кои што можат да ги решат овие задачи.

## Детали за имплементација

Вие треба да ја имплементирате следната процедура:

```
void construct_instructions(int s, int n, int k, int q)
```

- $s$: видот на задачата.
- $n$: бројот на цели броеви во влезот.
- $k$: бројот на битови во секој влезен цел број.
- $q$: максимален број на дозволени инструкции.
- Оваа процедура е повикана точно еднаш и треба да конструира секвенца од инструкции кои ќе ја извршат бараната задача.

Оваа процедура треба да повика една или повеќе од следните процедури за да ја конструира секвенцата од инструкции.

```
void append_move(int t, int y)
void append_store(int t, bool[] v)
void append_and(int t, int x, int y)
void append_or(int t, int x, int y)
void append_xor(int t, int x, int y)
void append_not(int t, int x)
void append_left(int t, int x, int p)
void append_right(int t, int x, int p)
void append_add(int t, int x, int y)
```

- Секоја процедура додава $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ или $add(t, x, y)$ инструкција на програмата, соодветно.
- За сите релевантни инструкции, $t$, $x$, $y$ мора да е барем $0$ и најмногу $m-1$.
- За сите релевантни инструкции, $t$, $x$, $y$ не е задолжително да бидат попарно различни (т.е. може некои од нив да се еднакви).
- За $left$ и $right$ инструкциите, $p$ мора да биде барем $0$ и најмногу $b$.
- За $store$ инструкцијата, должината на $v$ мора да биде точно $b$.

Исто така може да ја повикувате следната процедура да ви помогне во тестирањето на вашето решение:

```
void append_print(int t)
```

- Повиците на оваа процедура ќе бидат игнорирани за време на оценувањето на вашето решение.
- Во пример-оценувачот, оваа процедура додава $print(t)$ операција на програмата.
- Кога пример-оценувачот ќе сретне $print(t)$ операција за време на извршувањето на програмата, тој печати $n$ $k$-битни цели броеви формирани од првите $n \cdot k$ битови од регистарот $t$ (поголеднете го делот "Пример Оценувач" за детали).
- $t$ мора да задоволува $0 \leq t \leq m-1$.
- Било кој од повиците на оваа процедура не го зголемува бројот на конструирани инструкции.

По додавањето на последната инструкција, `construct_instructions` треба да заврши со извршување. Програмата потоа е евалуирана на неколку тест примери, секој од кој специфицира влез кој се состои од $n$ $k$-битни цели броеви $a[0], a[1], \ldots, a[n-1]$. Вашето решение е оценето како точно за даден тест случај ако излезот од програмата $c[0], c[1], \ldots, c[n-1]$ за дадениот влез ги задоволува следните услови:

- Ако $s = 0$, $c[0]$ треба да е најмалиот број од броевите $a[0], a[1], \ldots, a[n-1]$.
- Ако $s = 1$, за секој $i$ ($0 \leq i \leq n-1$), $c[i]$ треба да биде $1+i$-тиот најмал број од броевите $a[0], a[1], \ldots, a[n-1]$.

Оценувањето на вашето решение може да резултира со некој од следните error пораки:

- `Invalid index`: неточен (можеби негативен) индекс за регистар бил предаден како параметар $t$, $x$ or $y$ за некој од повиците на процедурите.
- `Value to store is not b bits long`: Должината на $v$ дадена на `append_store` не изнесува $b$.
- `Invalid shift value`: вредноста $p$ дадена на `append_left` или `append_right` не е помеѓу $0$ и $b$ вклучително.
- `Too many instructions`: вашата процедура се обидела да конструира програма со повеќе од $q$ инструкции.

# Примери

## Пример 1

Suppose $s = 0$, $n = 2$, $k = 1$, $q = 1000$. There are two input integers $a[0]$ and $a[1]$, each having $k = 1$ bit. Before the program is executed, $r[0][0] = a[0]$ and $r[0][1] = a[1]$. All other bits in the processor are set to $0$. After all the instructions in the program are executed, we need to have $c[0] = r[0][0] = \min(a[0], a[1])$, which is the minimum of $a[0]$ and $a[1]$.

There are only 4 possible inputs to the program:

- Case $1$: $a[0] = 0, a[1] = 0$
- Case $2$: $a[0] = 0, a[1] = 1$

- Case $3$: $a[0] = 1, a[1] = 0$
- Case $4$: $a[0] = 1, a[1] = 1$

We can notice that for all 4 cases, $\min(a[0], a[1])$ is equal to the bitwise-AND of $a[0]$ and $a[1]$. Therefore, a possible solution is to construct a program by making the following calls:

1. `append_move(1, 0)`, which appends an instruction to copy $r[0]$ to $r[1]$.
2. `append_right(1, 1, 1)`, which appends an instruction that takes all bits in $r[1]$, shifts them to the right by $1$ bit, and then stores the result back in $r[1]$. Since each integer is $1$-bit long, this results in $r[1][0]$ being equal to $a[1]$.
3. `append_and(0, 0, 1)`, which appends an instruction to take the bitwise-AND of $r[0]$ and $r[1]$, then store the result in $r[0]$. After this instruction is executed, $r[0][0]$ is set to the bitwise-AND of $r[0][0]$ and $r[1][0]$, which is equal to the bitwise-AND of $a[0]$ and $a[1]$, as desired.

### Пример 2

Suppose $s = 1,\ n = 2,\ k = 1,\ q = 1000$. As with the earlier example, there are only 4 possible inputs to the program. For all 4 cases, $\min(a[0], a[1])$ is the bitwise-AND of $a[0]$ and $a[1]$, and $\max(a[0], a[1])$ is the bitwise-OR of $a[0]$ and $a[1]$. A possible solution is to make the following calls:

1. `append_move(1,0)`
2. `append_right(1,1,1)`
3. `append_and(2,0,1)`
4. `append_or(3,0,1)`
5. `append_left(3,3,1)`
6. `append_or(0,2,3)`

After executing these instructions, $c[0] = r[0][0]$ contains $\min(a[0], a[1])$, and $c[1] = r[0][1]$ contains $\max(a[0], a[1])$, which sorts the input.

## Ограничувања

- $m = 100$
- $b = 2000$
- $0 \leq s \leq 1$
- $2 \leq n \leq 100$
- $1 \leq k \leq 10$
- $q \leq 4000$
- $0 \leq a[i] \leq 2^k - 1$ (за сите $0 \leq i \leq n - 1$)

## Подзадачи

1. (10 поени) $s = 0, n = 2, k \leq 2, q = 1000$
2. (11 поени) $s = 0, n = 2, k \leq 2, q = 20$
3. (12 поени) $s = 0, q = 4000$

4. (25 поени) $s = 0, q = 150$
5. (13 поени) $s = 1, n \leq 10, q = 4000$
6. (29 поени) $s = 1, q = 4000$

## Оценувач

Дадениот оценувач го чита влезот во следниот формат:

- ред $1$: $s\ n\ k\ q$

Ова е проследено од неколку редови, секој од кој опишува еден тест пример. Секој тест пример треба да е во следниот формат:

- $a[0]\ a[1]\ \ldots\ a[n-1]$

и опишува тест пример чиј што влез се состои од $n$ цели броеви $a[0], a[1], ..., a[n-1]$. После сите тест примери треба да следи еден ред кој што го содржи единствено бројот $-1$.

Дадениот оценувач најпрво го прави повикот `construct_instructions(s, n, k, q)`. Ако овој повик ги прекршува некои од ограничувањата наведени во текстот на задачата, тогаш дадениот оценувач печати некои од грешките наведени во делот "Детали за имплементација" и завршува со извршување. Инаку, дадениот оценувач прво ја печати програмата конструирана од `construct_instructions(s, n, k, q)`. За $store$ инструкциите, $v$ е испечатено од индекс $0$ до индекс $b-1$.

Потоа, дадениот оценувач ги процесира тест примерите во редослед. За секој тест пример, ја извршува конструираната програма со влезот од тест примерот.

За секоја $print(t)$ операција, нека $d[0], d[1], \ldots, d[n-1]$ е секвенца од цели броеви, такви што за секој $i$ ($0 \leq i \leq n-1$), $d[i]$ е целобројната вредност на секвенцата од битови од $i \cdot k$ до $(i+1) \cdot k - 1$ од регистарот $t$ (во моментот кога операцијата е извршена). Оценувачот ја печати оваа секвенца во следниот формат: `register` $t$: $d[0]\ d[1]\ \ldots\ d[n-1]$.

Штом сите инструкции ќе бидат извршени, дадениот оценувач го печати излезот на програмата.

Ако $s = 0$, излезот на дадениот оценувач за секој тест пример е во следниот формат:

- $c[0]$.

Ако $s = 1$, излезот на дадениот оценувач за секој тест пример е во следниот формат:

- $c[0]\ c[1]\ \ldots\ c[n-1]$.

После извршувањето на сите тест примери, оценувачот печати `number of instructions:` $X$ каде $X$ е бројот на инструкции во вашата програма.