Курсов проект за Машини с неограничени регистри

Калин Георгиев
kalin@fmi.uni-sofia.bg

27 октомври 2019 г.

1 Дефиниция на МНР и примери

Дефиницията на Машина с неограничени регистри по-долу е взаимствана от учебника [1] *А. Дичев, И. Сосков, "Теория на програмите", Издателство на СУ, София, 1998.*

"Машина с неограничени регистри" (или МНР) наричаме абстрактна машина, разполагаща с неограничена памет. Паметта на машината се представя с безкрайна редица от естествени числа $m[0], m[1], \ldots$, където $m[i] \in \mathcal{N}$. Елементите $m[i]$ на редицата наричаме "клетки" на паметта на машината, а числото i наричаме "адрес" на клетката $m[i]$.

МНР разполага с набор от инструкции за работа с паметта. Всяка инструкция получава един или повече параметри (операнди) и може да предизвика промяна в стойността на някоя от клетките на паметта. Инструкциите на МНР за работа с паметта са:

1) `ZERO n`: Записва стойността 0 в клетката с адрес n
2) `INC n`: Увеличава с единица стойността, записана в клетката с адрес n
3) `MOVE x y`: Присвоява на клетката с адрес y стойността на клетката с адрес x

"Програма" за МНР наричаме всяка последователност от инструкции на МНР и съответните им операнди. Всяка инструкция от програмата индексираме с поредния ѝ номер. Изпълнението на програмата започва от първата инструкция и преминава през всички инструкции последователно,

освен в някои случаи, описани по-долу. Изпълнението на програмата се прекратява след изпълнението на последната ѝ инструкция. Например, след изпълнението на следната програма:

```
0: ZERO 0
1: ZERO 1
2: ZERO 2
3: INC 1
4: INC 2
5: INC 2
```

Първите три клетки на машината ще имат стойност 0, 1, 2, независимо от началните им стойности.

Освен инструкциите за работа с паметта, МНР притежават и една инструкция за промяна на последователността на изпълнение на програмата:

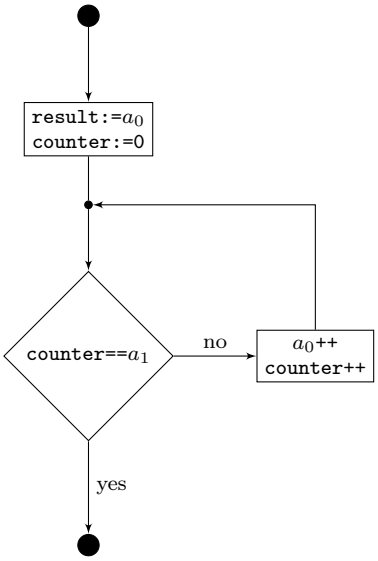
4) `JUMP z`: Изпълнението на програмата "прескача" и продължава от инструкцията с пореден номер z. Ако програмата има по-малко от $z+1$ инструкции, изпълнението ѝ се прекратява

5) `JUMP x y z`: Ако съдържанията на клетките x и y съвпадат, изпълнението на програмата "прескача" и продължава от инструкцията с пореден номер z. В противен случай, програмата продължава със следващата инструкция. Ако програмата има по-малко от $z+1$ инструкции, изпълнението ѝ се прекратява. Забележете, че горният вариант на инструкцията `JUMP` с един операнд (за безусловен преход) е частен случай на инструкцията `JUMP` с три операнда (за условен преход). Така `JUMP z` и може да се "симулира", например, с `JUMP 0 0 z`.

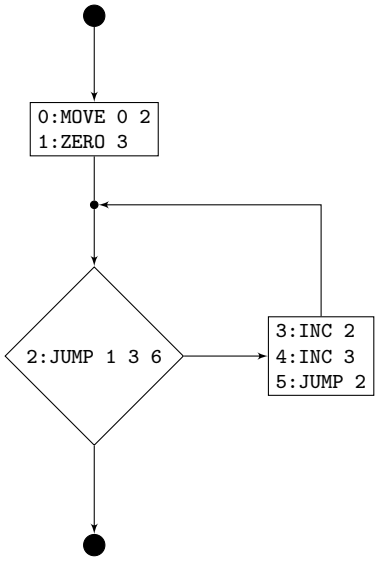
Например, нека изпълнението на следната програма започва при стойности на клетките на паметта $10, 0, 0, \ldots$:

```
0: JUMP 0 1 5
1: INC 1
2: INC 2
3: INC 2
4: JUMP 0
```

След приключване на програмата, първите три клетки на машината ще имат стойности 10, 10, 20.



(a) Програма за сумиране на числата a_0 и a_1 с използване само на операторите `=`, `==`, `++` и `if`.



(b) Програма за сумиране на клетките $m[0]$ и $m[1]$ с инструкции на МНР.

Фигура 1: Блок схеми на програма за сумиране на числа

Примери: На Фигура 1 (a) е показана блок схема на програма, използваща само операторите `=`, `==`, `++` и `if`, която намира в променливата `result` сумата на променливите a_0 и a_1. a_0 и a_1 считаме за дадени. Променливата `count` се инициализира с 0, а `result` - с a_0. В цикъл се добавя по една единица към `count` и `result` дотогава, докато `count` достигне стойността на a_1. По този начин, към `result` се добавят a_1 на брой единици, т.е. стойността ѝ се увеличава с a_1 спрямо налчалната ѝ стойност a_0.

На Фигура 1 (b) е показана същата програма, като операторите от първата са заменени със съответните им инструкции на МНР. Резултатът от програмата се получава в клетката $m[2]$, а за брояч се ползва клетката $m[3]$. На блок схемата са дадени поредните номера на инструкциите в окончателната програмата на МНР:

```
0:MOVE 0 2
1:ZERO 3
2:JUMP 1 3 6
3:INC 2
4:INC 3
5:JUMP 2
```

1.1 Примерни задачи за програми за МНР

1.1. Нека паметта на МНР е инициализирана с редицата $m, n, 0, 0, ...$ Да се напише програма на МНР, след изпълнението на която клетката с адрес 2 съдържа числото $m + n$.

1.2. Нека паметта на МНР е инициализирана с редицата $m, n, 0, 0, ...$ Да се напише програма на МНР, след изпълнението на която клетката с адрес 2 съдържа числото $m \times n$.

1.3. Нека паметта на МНР е инициализирана с редицата $m, n, 0, 0, ...$ Да се напише програма на МНР, след изпълнението на която клетката с адрес 2 съдържа числото 1 тогава и само тогава, когато $m > n$ и числото 0 във всички останали случаи.

2 Условие на проекта

"Програма за МНР" наричаме последователност от два вида оператори: *инструкции* и *команди*.

Инструкциите за МНР са от петте вида, описани по-горе: `ZERO x`, `INC n`, `MOVE x y`, `JUMP z`, `JUMP x y z`. Инструкциите са номерирани с последователни естествени числа спрямо реда им в програмата.

Командите на МНР започват със символа `/` и нямат поредни номера. Командите в програмата не влияят върху номерацията на инструкциите. Видовете команди, които могат да бъдат включвани в програмите за МНР, са описани по-долу.

Да се реализира интерпретатор за програми на МНР. Интепретаторът да работи в диалогов режим като приема команди от стандартния вход и ги изпълнява веднага след въвеждането им. Във всеки момент интерпретаторът поддържа в паметта "заредена програма", състояща се от всички въведени инструкции и команди на интерпретатора.

Интерпретаторът да поддържа следните команди:

2.1. `/zero x y`: Нулира клетките на паметта с адреси от `x` до `y` включително.

2.2. `/set x y`: Променя на `y` съдържанието на клетката с адрес `x`.

2.3. `/copy x y z`: Копира съдържанието на `z` последователни клетки, започващи от адрес `x` в съответните `z` последователни клетки, започващи от адрес `y`.

2.4. `/mem x y`: Извежда на стандартния изход съдържанието на клетките с адреси от `x` до `y` включително.

2.5. `/load <filename>`: Зарежда програма за МНР от текстов файл. Зареденaта до момента програма за МНР в паметта на интерпретатора се изтрива и се замества с новата програма, заредена от файла. Програмата не се изпълнява при зареждането на файла. Ако командата е част от текущо изпълняваща се програма, изпълнението на програмата се прекратява.

2.6. `/run`: Изпълнява заредената програма, като спазва последователността на инструкциите и командите и започва изпълнението от първия оператор на програмата. След изпълнението на даден оператор (инструкция или команда), различен от инструкцията `JUMP`, се преминава към изпълнението на следващия в последователността. Изпълнението на програмата се прекратява при опит за преминаване извън заредената последователност от оператори. Ако командата е част от текущо изпълняваща се програма, се преминава към изпълнението на първия оператор в програмата, без да се правят промени по паметта на машината.

2.7. `/call x`: Изпълнява заредената програма, като започва изпълнението от инструкцията с пореден номер `x`, а не от началото.

2.8. `/add <filename>`: Разширява заредената програма за МНР с оператори, прочетени от текстов файл.

Обхват на програма наричаме наричаме целочисления интервал $[a, b]$, където a е най-малкия адрес на клетка от паметта, който се използва в някоя инструкция на интерпретатора (но не и в командите!), а b е най-големия такъв адрес.

Ако преди изпълнението на командата `/add` в паметта има заредена програма с обхват $[a, b]$ и N на брой инструкции, а новата програма има обхват $[a', b']$, новата програма P да се преработи така, че:

- да има обхват $[a' + b + 1, b' + b + 1]$, т.е. всички адреси на клетки, срещащи се в инструкции на P, да се увеличат с числото $b + 1$;
- всички номера на инструкции, които се споменават в инструкции `JUMP` в P, да бъдат увеличени с N.

Операторите на P да се добавят последователно към края на заредената в паметта програма.

2.9. `/quote <string>`: Добавя нов оператор (инструкция или команда) `<string>` на края на заредената програма. Командата връща грешка, в случай, че `<string>` не е валиден оператор.

2.10. `/code`: Извежда на стандартния изход заредена в паметта програма.

2.11. `/comment <string>`: Няма ефект, служи за добавяне на коментар

Примерна програма:

Файл `fib.urm`:

```
/comment Програма за пресмятане на n-тото число на Fibonacci
/comment Числото n е записано в клетка с адрес 0
/comment Резултатът се записва в клетка с адрес 1
/comment Помощни клетки:
/comment Адрес 2: предишното число на Fibonacci
/comment Адрес 3: брояч, който се инкрементира до n
/comment Обхват: [0, 3]
/comment Брой инструкции: 10
/comment Нулиране на помощните клетки
ZERO 1
ZERO 2
ZERO 3
/comment Зареждане на подпрограма за добавяне на числа
/add add.urm
/comment Добавяне на оператор за връщане в главната програма
/comment при приключване на работа на подпрограмата
/quote JUMP 7
/comment Ако n = 0, край
JUMP 0 3 16
/comment Инициализация на числото на Fibonacci с номер 1 на 1 в клетка с адрес 1
INC 1
/comment Инициализация на брояча с 1
INC 3
/comment Ако n (записано на адрес 0) съвпада с брояча, край
JUMP 0 3 16
/comment Събиране на последните две числа на Fibonacci
/comment Подготовка на входа на подпрограмата add.urm
/copy 1 2 4
/comment Извикване на подпрограмата
/call 10
/comment Връщане от подпрограмата на инструкция с номер 7
/comment Запазване на текущото число на Fibonacci на адрес 1 като предишно на адрес 2
MOVE 1 2
/comment Копиране на новото число на Fibonacci, сметнато от подпрограмата, на адрес 1
/copy 6 1 1
/comment Увеличаване на брояча
INC 2
/comment Връщане в началото на цикъла
JUMP 3
```

Файл `add.urm`:

```
/comment Програма за събиране на две числа в клетки с адреси 0 и 1
/comment Резултатът се записва в клетка с адрес 2
/comment Помощна клетка с адрес 3: брояч, който се инкрементира до числото на адрес 1
/comment Обхват: [0, 3]
/comment Брой инструкции: 6
/comment Нулиране на брояча
ZERO 3
/comment Прехвърляне на стойността на клетката с адрес 0 в резултата на адрес 2
MOVE 0 2
/comment Проверка за край
JUMP 1 3 6
/comment Инкрементиране на брояча и резултата
INC 2
INC 3
/comment Връщане в началото на цикъла
JUMP 2
```

Примерно изпълнение от стандартния вход:

```
/load fib.urm
/set 0 9
/run
/mem 1 1
/comment Извежда се 34
/mem 0 6
/comment Извежда се 9 34 21 9 21 13 34 13
```

Внимание: За реализация на паметта на МНР да се използва "разреден масив" (*sparse array*) по алгоритъм, избран от вас.

Литература

[1] А. Дичев, И. Сосков, "Теория на програмите", Издателство на СУ, София, 1998