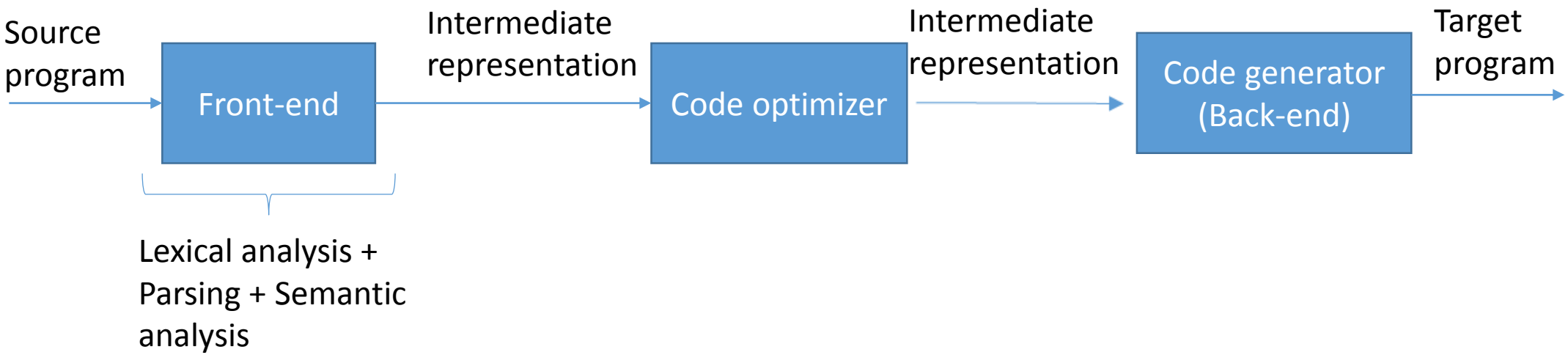
Генериране на код


Source program
Front-end
Intermediate representation
Code optimizer
Intermediate representation
Code generator (Back-end)
Target program
Lexical analysis + Parsing + Semantic analysis

Основни цели на генератора

- Избор на инструкции (instruction selection)
- Заемане на регистри (register allocation and assignment)
- Подреждане на инструкции (instruction ordering)

Цел 1: избор на инструкции

- Какви инструкции да бъдат избрани в съответствие със структурата на целевата машина (target machine)?

Цел 2: заемане на регистри

- Коя стойност в кой регистър да бъде записана (съхранена)?

Цел 3: подреждане на инструкции

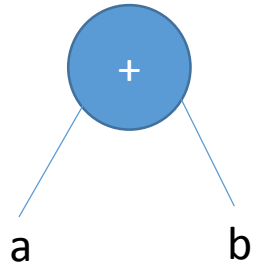
- В какъв ред да бъдат изпълнени генерираните инструкции?

Основни проблеми при проектиране на генератор на код

- Коректност
- Лесна реализация, коректност и поддръжка

Вход на генератора

- Междинна форма (intermediate representation) – синтактично дърво
- Пример: синтактично дърво за израз **a+b**



Изходна програма – целева архитектура

- RISC (reduced instruction set computer)
  - Множество регистри
  - Триадресни инструкции
  - Просто адресиране
  - Ограничен брой прости инструкции
- CISC (complex instruction set computer)
  - Малко регистри
  - Двуадресни инструкции
  - Различни режими за адресиране
  - Сложни инструкции (с променлива дължина)

Изходна програма – целева архитектура

- Стекова машина (stack based)
  - Съхраняване на операнди в стек
  - Изпълнение на операциите с операндите във върха на стека
  - Съхраняване на указател към върха на стека в регистър

Избор на архитектура за генериране на код

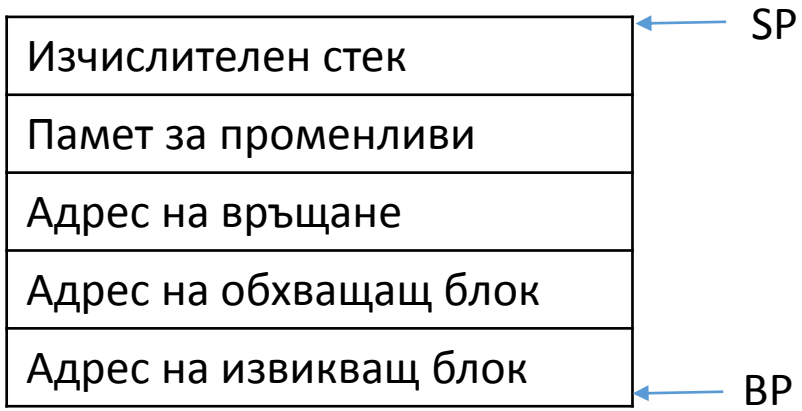
- Универсална стекова машина
  - Операндите се съхраняват в стек
  - Операциите се изпълняват с операндите в стека
- Регистри
  - Програмен брояч – адрес на текущата инструкция
  - Базов указател – начало на фрагмент в стека за текущ блок (главна програма или подпрограма)
  - Стеков указател – адрес на върха на стека (последната заета клетка)

Формат на команда

Код на операцията	Операнд 1	Операнд 2	Адрес на резултата

Стеков фрагмент

- За всеки блок (програма или подпрограма – процедура/функция) се генерира област в паметта (стеков фрагмент) със следната структура



Примерен набор от макрокоманди

- LOAD a – запис на съдържанието на клетка с адрес **a** във върха на стека
- LDADR a – запис на адреса на **a** във върха на стека
- STORE a – запис на стойността от върха на стека в клетка с адрес **a**
- ADD, SUB, MUL, DIV – прочитат двете най-горни клетки в стека, извършват действие, записват резултата в стека

Генериране на команди за аритметични изрази

- Примери

a+b*c

(a+b)*b

- **Как да се определи приоритет на операциите при сканиране на изречението отляво надясно?**

Премахване на скоби

(a+b)

+(a,b)

+ab -> префиксен запис

ab+ -> постфиксен запис – обратен полски запис (ОПЗ)

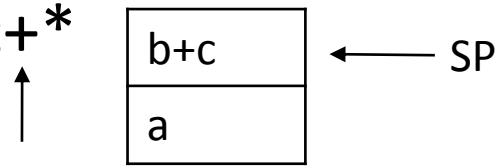
- Примери:

ab+c* -> (a+b)*c

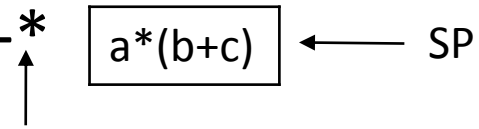
a+bc* -> ?

ab+cd*+e*fg+* -> ?

- abc+*



- abc+*



Пример за генериране на код

y:=a+b*(c+d)

ОПЗ -> yabcd+*+:=

y – операнд от адресен тип

a, b, c, d – операнд от стойностен тип

Компилиране в макрокоманди – работна програма

LDADR y

LOAD a

LOAD b

LOAD c

LOAD d

ADD -

MUL -

ADD -

STORE -

Изпълнение на командите в стека

- LDADR y

- LOAD a

- LOAD b

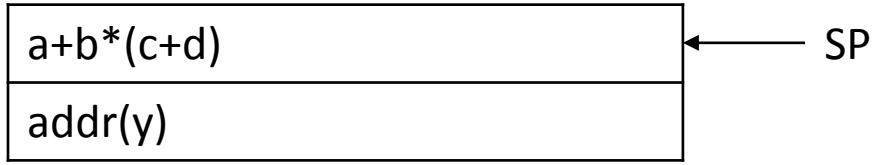
- LOAD c

- LOAD d

- ADD

- MUL

- ADD



- STORE