Сегментирование в ПК. Префиксы замены сегмента

А.А. Вылиток

Оперативная память ПК имеет объем 2^{20} байтов (1 МБ). Байт — минимально адресуемая часть ОП, поэтому для явного указания адреса операнда, находящегося в ОП, необходимо 20 двоичных разрядов. Чтобы не указывать в командах 20-разрядный адрес целиком, и тем самым сократить размер команд (и, как следствие, машинных программ), в ПК принят способ адресации, названный *сегментированием*. Он состоит в следующем. Память условно делят на участки, называемые сегментами. В ПК *сегмент памяти* — это непрерывный участок ОП длиной не более 2^{16} байт, начинающийся по адресу кратному 16-ти. Абсолютный (физический) адрес ячейки памяти из некоторого сегмента можно представить в виде $A = B + ofs$, где B — адрес начала сегмента (*база*), *ofs* — адрес (номер) ячейки, отсчитанный от начала сегмента (*смещение*). Так как размер сегмента не превосходит 2^{16} байт, для задания смещения достаточно 16-ти разрядов. Для задания базы также достаточно 16-ти разрядов, поскольку адрес начала сегмента кратен 16-ти, т.е. последние четыре бита этого адреса всегда нулевые – их можно не записывать, но учитывать при вычислении абсолютного адреса. Адрес начала сегмента, деленный на 16, называется *номером сегмента*. Если N — номер сегмента, то формула для представления абсолютного адреса приобретает вид: $A = N \cdot 16 + ofs$. Для хранения номеров сегментов в ЦП предусмотрены четыре специальных регистра (16-разрядных): *CS*, *SS*, *DS*, *ES*. Это так называемые *сегментные регистры*.

Напомним, что в команде ПК может быть явно задано не более одного операнда, находящегося в ОП (не существует формата команд «память-память»). При выполнении команды абсолютный адрес её операнда вычисляется в два этапа. Сначала вычисляется так называемый *исполнительный адрес*: содержимое адресного поля команды складывается с содержимым модификаторов (если они используются в данной команде), полученный результат берется по модулю 2^{16} [(*)].

Далее абсолютный адрес вычисляется по формуле:

$$A_{абс} = (A_{исп} + SR \cdot 16) \bmod 2^{20},$$

где $A_{исп}$ — исполнительный адрес, SR — значение сегментного регистра. Какой именно сегментный регистр используется, ведь он не указывается в команде? Ответ таков. Каждая команда, содержащая адресный операнд, априори подразумевает использование конкретного сегментного регистра: *SS* или *DS*. Это так называемые сегментные регистры *по умолчанию*. В этом отношении команды как бы делятся на два класса: те, в которых есть модификация по *BP*, используют по умолчанию *SS*, остальные — *DS*. Если мы хотим, чтобы в данной команде был использован другой сегментный регистр, например *ES*, или *CS*, или *SS* вместо *DS*, или *DS* вместо *SS*, то перед данной командой нужно поместить специальную однобайтовую команду, называемую *префиксом замены сегмента*. Таких команд всего четыре — по одной на каждый сегментный регистр. Для сегментного регистра *ES* машинный код префикса замены равен 26h, для *SS* — 36h, для *CS* — 2Eh, для *DS* — 3Eh.

Сделаем замечание по поводу команд перехода. Как известно, в ПК адрес очередной команды, подлежащей выполнению, определяется парой регистров *CS*:*IP*. Абсолютный адрес этой команды вычисляется по формуле: $(IP + CS \cdot 16) \bmod 2^{20}$.

---

[*] Адресное поле может отсутствовать (тогда обязательно присутствуют модификаторы); модификаторов может быть один или два. К регистрам-модификаторам в ПК относятся *BP*, *BX*, *SI*, *DI*. Если в команде указана модификация адреса по двум регистрам, то из их названий можно составить слово «BI»; например, *SI* и *BX* — допустимая пара модификаторов.

Передача управления осуществляется посредством изменения значений *IP* и *CS*. В начале выполнения любой команды значение IP автоматически увеличивается на длину исполняемой команды, так что происходит переход к следующей команде, и команды выполняются последовательно. Чтобы изменить порядок выполнения, используются команды перехода (отметим, что с помощью пересылок регистры *IP* и *CS* менять нельзя). Команды, которые изменяют только IP, называются командами *внутрисегментного (близкого)* перехода. Команды, которые изменяют *IP* и *CS*, называются командами *межсегментного (дальнего)* перехода. Если информация о новом значении *IP* и, в случае межсегментного перехода — *CS*, присутствует непосредственно в команде, то это команда *прямого перехода*, иначе — *косвенного*. В случае косвенного перехода информация о новом значении *IP* (и *CS*) извлекается из ОП или регистра, а сама команда лишь определяет то место, где хранится новое значение.

Команды прямого внутрисегментного перехода бывают *длинные* и *короткие*. Длинные состоят из трех байтов и, кроме кода операции, содержат новое значение (размером в слово) IP (т.е. смещение относительно начала сегмента, номер которого находится в CS). Получается, что в такой команде как бы есть адресное поле, но сегментирования не происходит, поскольку адрес в этой команде не указывает на операнд в ОП, а сам используется как непосредственный операнд. Короткие команды перехода состоят из двух байтов. Вместо смещения во втором байте указывается величина (от −128 до +127), которую следует прибавить к текущему значению *IP*, чтобы получить новое значение. Отметим также, что команды условного перехода бывают только короткими, все остальные (т.е. не короткие) переходы — безусловные.

В командах прямого межсегментного перехода кроме значения *IP* указывается еще и новое значение *CS* (вместе с кодом операции длина такой команды составляет 5 байт).

Что касается команд косвенного перехода (в них указывается адрес в ОП, по которому находятся значения *IP* и, в межсегментном случае, *CS*), то при их выполнении сегментирование осуществляется описанным выше способом как для любой другой команды, содержащей адрес операнда в ОП. Бывают еще команды косвенного перехода по регистру. В них, разумеется, сегментирование отсутствует, так как адрес перехода берется из регистра, а не из ОП.

Для того, чтобы сегментирование происходило корректно, сегментные регистры должны быть настроены на соответствующие номера сегментов. Регистр *CS* всегда настраивается автоматически операционной средой перед запуском программы. Для *SS* также возможна автоматическая настройка. Остальные должны быть настроены в программе с помощью команд, загружающих номер сегмента в сегментный регистр. (Непосредственный операнд напрямую нельзя записать в сегментный регистр, поэтому номер сегмента сначала записывается в регистр общего назначения, например *AX*, а затем из него пересылается в сегментный регистр.) В процессе выполнения программы можно перенастроить сегментные регистры на другие сегменты, изменив нужным образом значения регистров (кроме регистра *CS* — в него нельзя переслать значение – он меняется только в результате межсегментных переходов).

На языке ассемблера сегменты памяти изображаются с помощью *программных сегментов*. Программный сегмент — это последовательность предложений ЯА, ограниченная директивами *SEGMENT* и *ENDS*. Каждый программный сегмент имеет имя, оно указывается дважды – в открывающей и закрывающей сегмент директивах:

```
<имя сегмента> SEGMENT <параметры>
        <предложение>
                …
        <предложение>
<имя сегмента> ENDS
```

Программные сегменты с одним и тем же именем объединяются в один сегмент памяти в порядке их расположения в программе. Объём сегмента не должен превышать 64K, иначе ассемблер зафиксирует ошибку. Имя сегмента в ЯА относится к константным выражениям, его значением является номер соответствующего сегмента памяти.

Предложения ЯА, задающие команды и данные, не могут находиться вне программного сегмента. Программа на ЯА представляет собой последовательность программных сегментов. Обычно программа на ЯА состоит, по меньшей мере, из трёх сегментов: сегмента стека, сегмента данных и сегмента команд. В конце программы стоит директива *END <точка входа>*, где *<точка входа>* — имя метки, стоящей перед той командой, с которой должно начаться выполнение программы. При запуске программы сегментный регистр *CS* автоматически настраивается на сегмент, содержащий точка входа (*IP* соответственно настраивается на смещение точки входа относительно начала сегмента).

При описании сегмента стека в директиве *SEGMENT* указывается параметр *STACK*. Это позволяет не заботиться о настройке сегментного регистра *SS* — в этом случае при запуске программы он настраивается автоматически. Для безошибочной трансляции сегмента команд с именем *<имя>* перед ним должна стоять директива *ASSUME CS*:*<имя>*. В противном случае, встретив первую же метку, ассемблер сообщит об ошибке.

Вообще директива *ASSUME <SR>:<имя программного сегмента>* сообщает ассемблеру, что переменные, расположенные в данном программном сегменте, должны сегментироваться по регистру <*S*R>, что позволяет не указывать явно этот регистр в команде. Ассемблер «помнит» о соответствии между регистром *<SR>* и сегментом вплоть до следующей директивы, устанавливающей новое соответствие для *<SR>*. В одной директиве *ASSUME* через запятую можно задать несколько соответствий. Можно также отменить уже установленные соответствия, не задавая новых. Это делается директивой *ASSUME NOTHING* (отменяет все ранее установленные соответствия между регистрами и сегментами) или *ASSUME <SR>:NOTHING* (отменяет соответствия, ранее установленные с данным сегментным регистром). Одному сегментному регистру в программе могут соответствовать (поочередно) несколько сегментов. Это естественно, так как сегментных регистров всего четыре, а сегментов может быть больше четырех. Отметим, что ASSUME не обеспечивает настройку сегментных регистров и их перенастройку с одного сегмента на другой. Настройка делается с помощью команд программы (автоматически настраиваются к началу выполнения программы только *CS* и *SS*).

Теперь подробно рассмотрим вопрос, как выбираются сегментные регистры при трансляции команд*.

В команде ЯА операнд, находящийся в ОП, может быть задан следующими способами:

(1) *<адресное выражение>*,
(2) *<SR>*:*<адресное выражение>*,
(3) *<SR>*:*<константное выражение>*,

---

* Напомним, что в некоторых командах вопрос о выборе сегментного регистра не стоит. Например, в команде прямого перехода: *jmp <метка>*. Действие этой команды таково: если метка близкая (типа *near*), то изменяется только *IP* (он станет равным адресу (смещению) метки), если метка дальняя (типа *far*), то соответственно изменяются и *CS*, и *IP*. Абсолютный адрес не вычисляется.

Другой пример — *movsb*. В этой команде схема вычисления абсолютных адресов жестко зафиксирована: байт по адресу $(DS \cdot 16 + SI) \bmod 2^{20}$ пересылается на место байта по адресу $(ES \cdot 16 + DI) \bmod 2^{20}$, и нет возможности использовать для задания адресов другие регистры

где <*SR*> — это сегментный регистр (*CS*, *SS*, *DS* или *ES*). Адресные выражения в (1)-(2) и константное выражение в (3) задают исполнительный адрес ($A_{исп}$).

Для команд ЯА с явно заданным операндом в ОП возникает вопрос:

Какой именно сегментный **регистр** будет **использован** при вычислении абсолютного адреса операнда данной команды во время выполнения программы?

Если сегментный регистр указан в команде (способы (2)-(3)), то именно он и будет использован. Например, для операнда *DS*:0001h будет использован *DS*, для *ES*:*Y* будет использован *ES*.

Если сегментный регистр не указан в команде (способ (1)), то он определяется по следующему алгоритму.

Алгоритм

Вход : адресное выражение — операнд команды из программы на ЯА.

Выход: сегментный регистр *SR*, который будет использован при вычислении абсолютного адреса операнда.

Введём обозначения:

- *expr* — адресное выражение, задающее операнд команды, находящийся в памяти *Замечание*. Адресное выражение считаем заданным в «канонической форме»: <*имя*> + [<*модификаторы*>] + <*константное выражение*>. Каждая из трех частей (имя переменной, модификаторы, константное выражение) может отсутствовать, но обязательно наличие хотя бы одой из двух первый частей, т.е. имени или модификаторов.
- *name* — имя, т.е. первая часть *expr* (если она присутствует)
- *R* — множество сегментных регистров, связанных к моменту трансляции данной команды директивой *ASSUME* с программным сегментом, содержащим описание имени name. *R* может быть пустым ($R = \varnothing$).
- *M* — множество модификаторов, содержащихся в *expr*. Может быть пустым ($M = \varnothing$).

Пример.

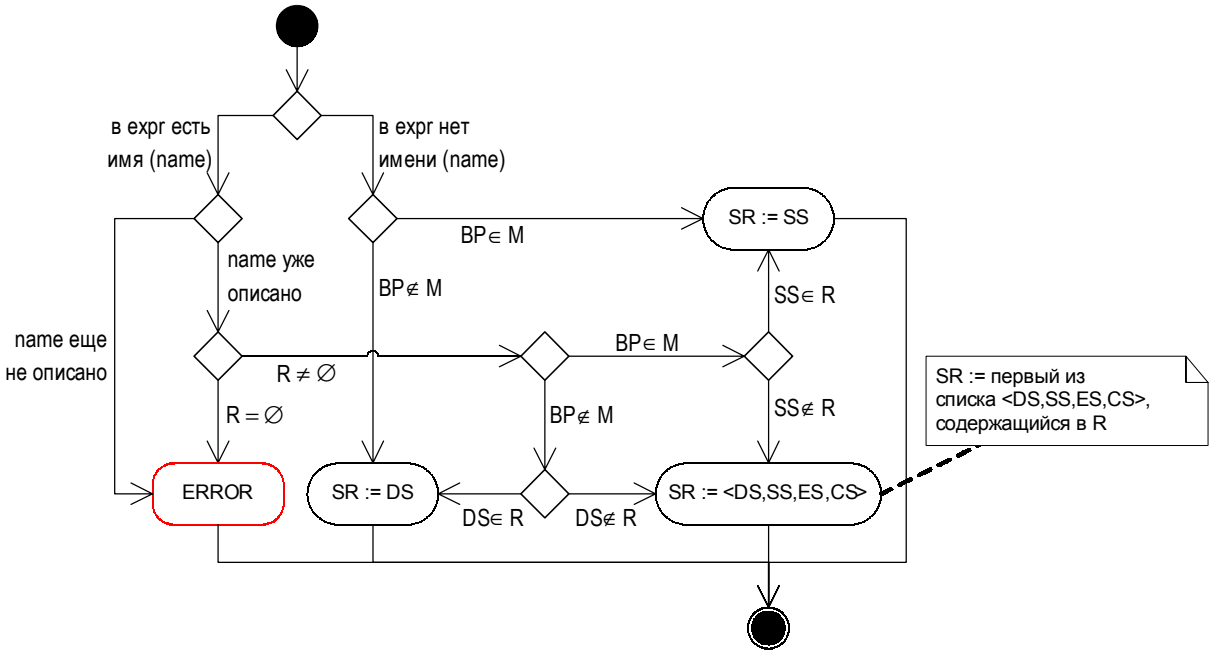
```
data segment
    x dw ?,?
data ends

code segment
    assume cs:code
    …
    mov x, 100h         ; R=∅,       expr=name=x,           M=∅,

    assume cs:code, ds:data
    …
    mov x[bx], 200h     ; R={ds},    expr=x[bx],    name=x, M={bx}

    assume cs:code, ds:data, es:data
    …
    mov x[bx,si], 300h ; R={ds,es}, expr=x[bx,si], name=x, M={bx,si}
code ends
```

Метод



Замечание. Если *name* описано ниже транслируемой команды, есть два случая, когда трансляция все же пройдет успешно: (1) $BP \notin M$ и $DS \in R$; (2) $BP \in M$ и $SS \in R$.

Итак, мы теперь знаем, как ассемблер определяет сегментный регистр, который должен использоваться при выполнении данной команды.

Остался последний вопрос: если в команде должен использоваться сегментный регистр *SR*, **вставит ли** ассемблер при трансляции этой команды соответствующий **префикс замены** сегмента? Ответ дает следующая таблица.

Используемый сегментный регистр	$BP \in M$?	Будет ли вставлен префикс
DS	да	да
	нет	нет
ES	да	да
	нет	да
CS	да	да
	нет	да
SS	да	нет
	нет	да

Пример.

```
s segment stack
    a dw ?
      dw 128 dup (?)
s ends

d1  segment
    b dw ?
d1  ends

d2  segment
    c dw ?
d2  ends


code segment
    assume ss:s, ds:d1, cs:code, es:d2
```

```
    d dw ?

start:
    mov ax,d1
    mov ds,ax

    ; команды программы
    ...
    mov ax, b[bp] ; вставит префикс ds
     ...
    mov ax, a     ; вставит префикс ss
     ...
    mov ax, c     ; вставит префикс es
     ...
    mov ax, b     ; не вставит префикс, используется ds
     ...
    mov ax, d     ; вставит префикс cs
     ...
fin:
    finish
code ends
end start
```