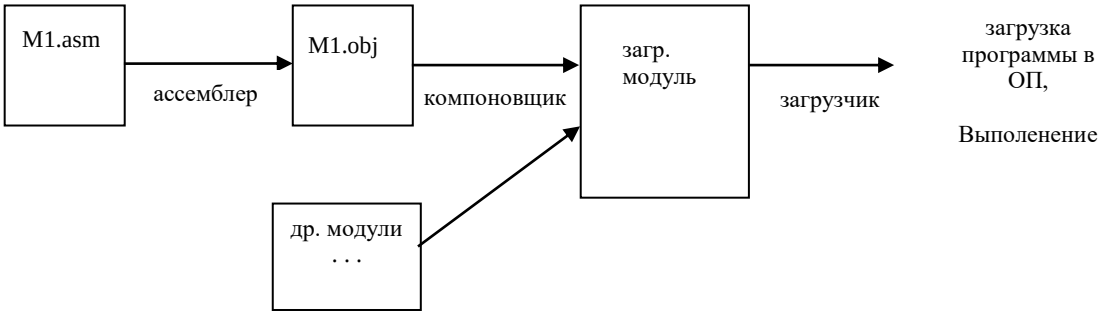
Тема: работа ассемблера, компоновщика, загрузчика.

Процесс обработки и выполнения программы:



§4. Работа ассемблера.

П 1. Суть ассемблирования

1. константы → в машинное представление
2. размещение переменных в ОП
3. трансляция команд
   а) КОП → код машинной операции
   б) операнды
      регистр – кодируется в команде (аналогия с УМ-Р)
      переменная – заменяется на адрес (как в УМ)
      константа – явно выписывается в команде

Проблема **ссылок вперёд**:
а) выбор машинного КОПа (работа с байтами/словами?)
б) какой адрес записывать в поле операнда машинной команды

⇒ **два прохода** ассемблера.
Цель первого прохода – собрать информацию об именах.
Цель второго прохода – построить объектный код.

П 2. Таблицы ассемблера.

Встроенные в ассемблер

а) Таблица директив

имя директивы	**процедура обработки**
EQU	адрес проц. EQU
DB	адрес проц. DB
…	…

б) Таблица мнемокодов

мнемокод	**op1**	**op2**	**КОП**	**размер**
ADD	AL	i	04	2
ADD	BL	i	80C3	3
ADD	EAX	m	0305	6
MOV	EAX	i	B8	5
CALL	метка		E8	5
RET			C3	1
…	…	…	…	…

Мнемокод, операнды ⇒ КОП, длина команды.
Обнаруживаемые ошибки: несуществующий мнемокод, недопустимое сочетание операндов или недопустимый операнд.

Строятся во время трансляции программы:

в) Таблица имён (ТИ)

имя	**тип**	**значение**	**сегмент**	**длина**	**атрибуты**
n	abs	70	-	-	
x	byte	0	DATA	5	
start	near	0	CODE	-	
…	…	…	…		

г) Таблица сегментов (ТС)

имя	**размер**
DATA	
CODE	

Рассматриваемый пример

Программа состоит из модулей M1(головной) и M2(вспомогательный).

Модуль M1.asm

```
include console.inc
EXTRN k:abs, P:near, x:dword
; имя процедуры P
.CODE
start:
    mov EAX,k
    call P
    add EAX, x
;  exit
END start
```

Модуль M2.asm

```
.586
.model flat,stdcall
option casemap:none
PUBLIC k, P, x
  k = 3
.DATA
  a dd 0
  x dd 6
  Y dd x
.CODE
  P PROC
    ADD EAX, Y
    RET
  P ENDP
end
```

В начале модуля M2.asm стоят директивы, управляющие режимом трансляции. В модуле M1.asm эти директивы входят в файл console.inc, вместе с другими директивами, которые вставляют в модуль M1.asm описания макросов (exit) и подключают библиотеки процедур, используемые в макросах.

Замечание: если хотите выполнить этот пример на компьютере – оттранслировать модули, объединить их и запустить exe-файл – нужно в модуле M1 заменить имя процедуры P на имя P@0 (так компилятор преобразует общие имена процедур) и убрать ';' перед командой exit.

Имена, описанные в сегментах (переменные, процедуры, метки) называются относительными (т.к. их значения – адреса, отсчитанные относительно начала сегмента) или перемещаемыми (т.к. в машинной программе, загруженной в ОП, их значения зависят от расположения сегментов в ОП). Такие имена в ТИ помечаются атрибутом R (relative), для них в ТИ записывается информация о местах в программе, где они используются.

ПЗ. Первый проход ассемблера. Трансляция модуля M2.asm.

Цель - собрать информацию об именах и заполнить таблицу имён.
Данные: программа на ассемблере, таблица директив и таблица мнемокодов.
Результат: таблица имён, таблица сегментов.

Программа читается строка за строкой, каждая строка-предложение обрабатывается: заполняются таблицы, вычисляется новое значение счётчика размещения cp (адрес размещаемого байта в сегменте). При обработке команд для определения длины команды используется информация из таблицы мнемокодов и уже собранная информация об описании имени.

Программа:

.586		настройка режима
.model flat,stdcall		трансляции
option casemap:none		модуля
PUBLIC k, P, x		→ в ТИ заносятся имена, атрибут publ
k = 3		→ в ТИ значение k
.DATA		→ в ТС, ср:=0
a dd 0		→ в ТИ значение имени a=ср=0; ср:=ср+4 (=4)
x dd 6		→ в ТИ, ср:=ср+4 (=8)
Y dd x		→ в ТИ, ср:=ср+4 (=12)
.CODE		→ в ТС размер DATA, ср:=0
P PROC		→ в ТИ тип и значение имени P
ADD EAX, Y		→ ср:= ср+6 (по табл. мнемокодов)
RET		→ ср:= ср+1
P ENDP		
end		→ размер CODE в ТС (=ср)
		конец первого прохода

ТИ

имя	тип	значение	сегмент	атрибуты	используется
k	number	3		publ	
P	near	0	CODE	publ,R	
x	dword	4	DATA	publ,R	DATA:8
a	dword	0	DATA	R	
Y	dword	8	DATA	R	CODE:2

ТС

имя	адрес нач. в модуле	размер
DATA	0000	0Ch
CODE	000Ch	7

На первом проходе обнаруживаются ошибки: повторное описание имени.

П4. Второй проход ассемблера.

Цель - построение объектного кода.
Данные: программа на ассемблере, таблица имён.
Результат: объектный код, таблицы для компоновщика.

Ассемблер просматривает текст программы с самого начала второй раз.

.586		адрес	Объектный код
.model flat,stdcall			
option casemap:none			
PUBLIC k, P, x			
k = 3			
.DATA	ср:=0		
a dd 0	ср:=ср+4 (=4)	0	0000 0000
x dd 6	ср:=ср+4 (=8)	4	0000 0006
Y dd x	ср:=ср+4 (=12), Адр. исп. x →в ТИ	8	0000 0004
.CODE	ср:=0	C	
P PROC			
ADD EAX, Y	ср:= ср+6, Адр. исп. Y →в ТИ	0	0305 0000 0008
RET	ср:= ср+1	6	C3
P ENDP			
end	конец трансляции		

Ошибки, обнаруживаемые на втором проходе: неописанное имя.

Замечания

1) Требования MASM <= принципы работы ассемблера

КВ1 dup (КВ2)

↑ ↑

без ссылок вперёд (исп-ся на 1 проходе)

допустимы ссылки вперёд (исп-ся на 2 проходе)

2) Ассемблер не может построить полную программу: не известны значения внешних имён.

Результаты трансляции модуля M1.asm

Таблицы

ТИ

имя	тип	значение	сегмент	атрибуты	используется
k	number			extrn	CODE:1
P	near			extrn	CODE:6
x	dword			extrn	CODE:0Ch
start	near	0	CODE	т.входа	

ТС

имя	адрес нач. в модуле	размер
CODE	0000	10h

Объектный код

include console.inc		адрес	Объектный код
EXTRN k:abs, P:proc, x:dword			
; имя процедуры P			
.CODE			
start:			
mov EAX,k		0	B8 ---- ----
call P		5	E8 ---- ----
add EAX, x		A	0305 ---- ----
; exit		10	
END start			

§5. Компоновка и загрузка.

Задача компоновщика– объединить несколько объектных модулей в одну программу:
1) Разрешение внешних связей.
2) Объединение модулей.
3) Информация для загрузчика (таблица перемещаемых адресов, точка входа).

Работа компоновщика.
1. Объединение таблиц имён: слить в одну таблицу.

ТИ

	имя	тип	значение	сегмент	атрибуты	используется
M1	k	number			extrn	CODE.M1:1
	P	near	0	CODE.M2	extrn	CODE.M1:6
	x	dword			extrn	CODE.M1:0Ch
	start	near	0	CODE.M1	т.входа	
M2	k	number	3		publ	
	P	near	0	CODE.M2	publ,R	
	x	dword	4	DATA	publ,R	DATA:8
	a	dword	0	DATA	R	
	Y	dword	8	DATA	R	CODE.M2:2

Склеить строки таблиц разных модулей, относящиеся к одному имени.

имя	тип	значение	сегмент	атрибуты	используется
k	number	3			CODE.M1:1
P	near	0	CODE.M2	R	CODE.M1:6
x	dword	4	DATA	R	CODE.M1:0Ch, DATA:8
start	near	0	CODE.M1	т.входа	
Y	dword	8	DATA	R	CODE.M2:2

2. Объединение сегментов CODE модулей M1 и M2.
В объединённом модуле за байтами сегмента CODE модуля M1 будут записаны байты сегмента CODE модуля M2 => смещения всех имён из CODE.M2 увеличатся на размер CODE.M1 – на 10h.

имя	тип	значение	сегмент	атрибуты	используется
k	number	3			CODE.M1:1
P	near	0 +10h =10	CODE.M2	R	CODE.M1:6
x	dword	4	DATA	R	CODE.M1:0Ch, DATA:8
start	near	0	CODE.M1	т.входа	
Y	dword	8	DATA	R	CODE.M2:2+10h = 12h

3. Объединяем все сегменты в единый модуль.

За сегментом кода приписываем сегмент DATA. В разных моделях памяти требования на адреса, по которым могут начинаться сегменты, разные. Договоримся, что у нас адрес начала сегмента кратен 10h. Длина сегмента CODE 10h + 7h = 17h, следовательно, сегмент данных будет начинаться по адресу 20h. Адреса переменных сегмента данных теперь будут отсчитываться не от начала сегмента DATA, а от начала программы, значит, все смещения увеличатся на 20h.

имя	тип	значение		сегмент	атр.	используется
k	number	3				CODE:1
P	near	10		CODE	R	CODE:6
x	dword	4+20h=24h		DATA	R	CODE:0Ch, DATA:8+20h=28h
start	near	0		CODE	т.вх.	
Y	dword	8+20h=28h		DATA	R	CODE:12h

Корректируем относительные адреса в модуле и прописываем значения внешних имён. В результате выполнения шагов 2 и 3 получаем модуль

адрес	объектный код	комментарий
0	B8 0000 0003	MOV EAX,k M1
5	E8 0000 0010	CALL P
A	0305 0000 0024	ADD EAX, x
10	0305 0000 0028	ADD EAX, Y (P) M2
16	C3	RET
17	//////////////	
20	0000 0000	a
24	0000 0006	x
28	0000 0024	Y

4. Таблица перемещаемых адресов (ТПА) и точка входа – для загрузчика.

Удаляем из ТИ лишнюю информацию, оставляем вхождение перемещаемых (относительных) адресов и точку входа. Получаем

имя	атр.	используется
P	R	6
x	R	0Ch, 28h
Y	R	12h

Точка входа: `start 0`

Работа загрузчика.

Загрузчик – часть операционной системы. Он загружает модуль в ОП, выполняет необходимые настройки и запускает программу на счёт.

1) Размещение модуля в ОП.
Становится известным адрес первого байта модуля. Например, пусть это адрес 4000h. Все перемещаемые адреса в программе нужно увеличить на 4000h. По ТПА определяем, какие двойные слова нужно изменять:

	адрес	объектный код	комментарий
4000h	0	B8 0000 0003	MOV EAX,k M1
4005	5	E8 0000 0010+ 4000h	CALL P
400A	A	0305 0000 0024+ 4000h	ADD EAX, x
4010	10	0305 0000 0028+ 4000h	ADD EAX, Y (P) M2
4016	16	C3	RET
4017	17	//////////////	
4020	20	0000 0000	a
4024	24	0000 0006	x
4028	28	0000 0024+ 4000h	Y

Корректируем адрес точки входа: Точка входа: `start 0`+4000h

Получаем, наконец, полную готовую к счёту программу:

	объектный код	комментарий
4000	B8 0000 0003	MOV EAX,k M1
4005	E8 0000 4010	CALL P
400A	0305 0000 4024	ADD EAX, x
4010	0305 0000 4028	ADD EAX, Y (P) M2
4016	C3	RET
4017	//////////////	
4020	0000 0000	a
4024	0000 0006	x
4028	0000 4024	Y

3) Настройка стека: загружается начальное значение ESP, соответствующее пустому стеку.

4) Передача управления на точку входа: JMP 4000h