Операции с аккумуляторами и адресными регистрами

10

Обзор главы

10.1	Обзор	10–2
10.2	ENT и LEAVE	10–3
10.3	Инкрементирование и декрементирование	10–6
10.4	+AR1 и +AR2: Прибавление константы к адресному регистру 1 или адресному регистру 2	10–7

10.1. Обзор

В Вашем распоряжении имеются следующие операции для того, чтобы обрабатывать содержимое одного или нескольких аккумуляторов или адресных регистров:

Мнемоника	Операция	Значение
TAK	Обменять содержимое AKKU 1 с содержимым AKKU 2	Эта операция обменивает содержимое AKKU 1 с содержимым AKKU 2.
PUSH с двумя Akku	Копировать AKKU 1 в AKKU 2	Эта операция копирует содержимое AKKU 1 в AKKU 2.
POP с двумя Akku	Копировать AKKU 2 в AKKU 1	Эта операция копирует содержимое AKKU 2 в AKKU 1.
PUSH с четырьмя аккумуляторами	Копировать AKKU 3 в AKKU 4, AKKU 2 в AKKU 3, AKKU 1 в AKKU 2	Эта операция копирует содержимое AKKU 3 в AKKU 4, содержимое AKKU 2 в AKKU 3 и содержимое AKKU 1 в AKKU 2.
POP с четырьмя аккумуляторами	Копировать AKKU 2 в AKKU 1, AKKU 3 в AKKU 2, AKKU 4 в AKKU 3	Эта операция копирует содержимое AKKU 2 в AKKU 1, содержимое AKKU 3 в AKKU 2 и содержимое AKKU 4 в AKKU 3.
ENT	Войти в Akku–стек	Эта операция копирует содержимое AKKU 3 в AKKU 4 и содержимое AKKU 2 в AKKU 3.
LEAVE	Покинуть Akku–стек	Эта операция копирует содержимое AKKU 3 в AKKU 2 и содержимое AKKU 4 в AKKU 3.
INC	Инкрементировать AKKU 1	Эта операция увеличивает содержимое младшего байта в младшем слове AKKU 1 на 8-битную константу, которая задана в команде. Эта константа может находиться в диапазоне от 0 до 255.
DEC	Декрементировать AKKU 1	Эта операция уменьшает содержимое младшего байта в младшем слове AKKU 1 на 8-битную константу, которая задана в команде. Эта константа может находиться в диапазоне от 0 до 255.
+AR1, +AR2	Прибавить AKKU 1 к адресному регистру	Эта операция прибавляет содержимое младшего слова AKKU 1 к адресному регистру 1 или 2.
+AR1 P#байт.бит, +AR2 P#байт.бит	Прибавить константу к адресному регистру	Эта операция прибавляет константу к содержимому адресного регистра 1 или 2.
BLD	Bild-команда	Эта операция не выполняет никакой функции и не влияет на биты состояния. Эта операция важна только для программатора (PG), когда отображается программа. Операнд <число> является опознавательным номером операции BLD и создается программатором.
NOP 0 NOP 1	Пустая команда 0 Пустая команда 1	Эти операции не выполняют никаких действий и не оказывают влияния на содержимое слова состояния. Операции NOP 1 и NOP 0 нужны для обратного перевода. Код этих операций содержит комбинацию битов из 16 нулей или 16 единиц.

Подробную информацию о смене последовательности байтов в AKKU 1 возьмите из главы 18.3.

10.2. ENT и LEAVE

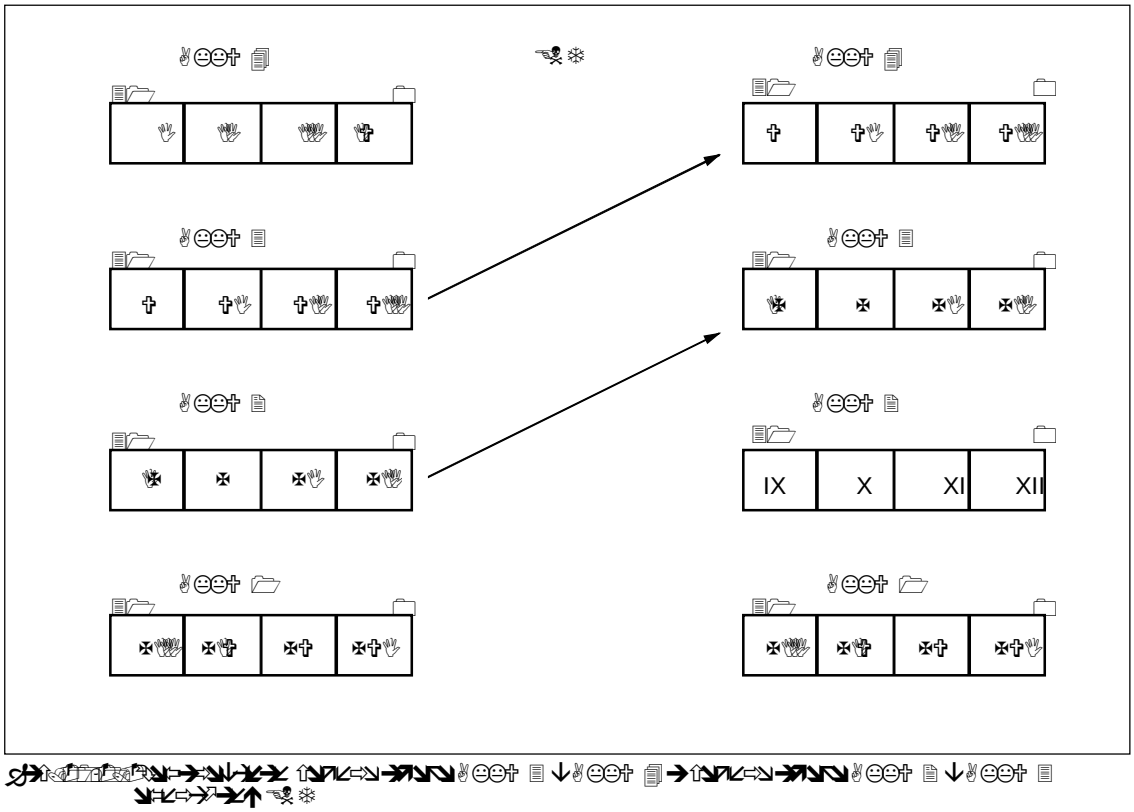
Описание

С помощью операций ENT (Enter Akku–Stack) и LEAVE (Leave Akku–Stack) Вы можете выполнять следующие функции:

- Операция ENT копирует содержимое AKKU 3 в AKKU 4 и содержимое AKKU 2 в AKKU 3. Если Вы программируете операцию ENT непосредственно перед операцией загрузки, то эта операция ENT сдвигает содержимое AKKU 2 и AKKU 3 глубже в стек.
- Операция LEAVE копирует содержимое AKKU 3 в AKKU 2 и содержимое AKKU 4 в AKKU 3. Если Вы программируете операцию LEAVE непосредственно перед операцией сдвига или циклического сдвига, связывающей аккумуляторы, то операция LEAVE функционирует как арифметическая операция.

ENT

Рисунок 10–1 показывает, как работает операция ENT.



LEAVE

Рисунок 10–2 показывает, как работает операция LEAVE.

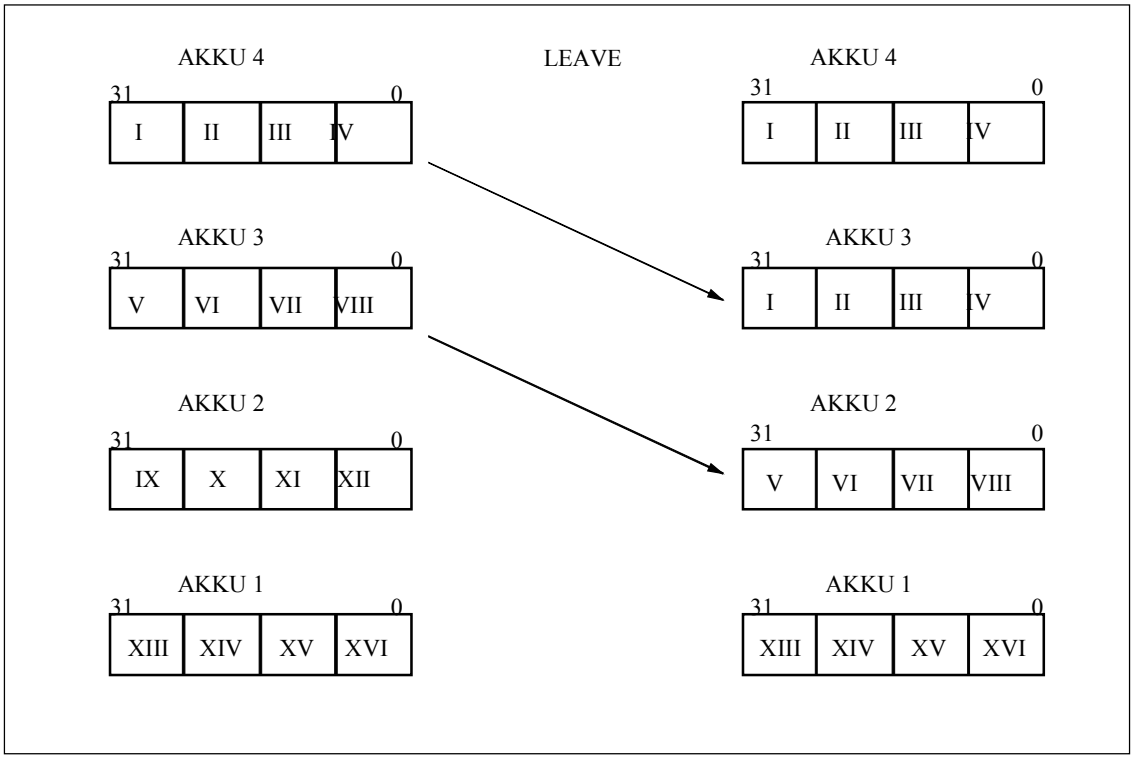


Рис. 10-2. Копирование содержимого AKKU 3 в AKKU 2 и содержимого AKKU 4 в AKKU 3 операцией LEAVE

Пример

Следующий фрагмент программы показывает использование операции ENT.

Нужно сложить числа с плавающей точкой, расположенные в словах данных DBD0 и DBD4. Сумму нужно разделить на разность чисел с плавающей точкой, расположенных в двойных словах данных DBD8 и DBD12.

$$DBD16 = \frac{DBD0 + DBD4}{DBD8 - DBD12}$$

Результат должен записываться в двойное слово данных DBD16.

Операция ENT служит в данном примере для того, чтобы сохранить в AKKU 3 промежуточный результат (DBD0+DBD4), который находится в AKKU 2. Команда вычитания (-R) копирует сохраненный в AKKU 3 промежуточный результат снова в AKKU 2 вслед за вычитанием.

AWL		Объяснение
L	**DBD0**	**Загрузить значение из двойного слова данных DBD0 в AKKU 1. (Это значение должно иметь формат числа с плавающей точкой.)**
L	**DBD4**	**Копировать значение из AKKU 1 в AKKU 2. Загрузить значение из двойного слова данных DBD4 в AKKU 1. (Это значение должно иметь формат числа с плавающей точкой.)**
+R		**Сложить содержимые AKKU 1 и AKKU 2 как числа с плавающей точкой (32 бита, IEEE–FP) и запомнить результат в AKKU 1. Копировать значение из AKKU 1 в AKKU 2.**
L	**DBD8**	**Загрузить значение из двойного слова данных DBD8 в AKKU 1. Копировать содержимое AKKU 3 в AKKU 4.**
ENT		**Копировать содержимое AKKU 2 (промежуточный результат) в AKKU 3. Копировать содержимое AKKU 1 в AKKU 2**
L	**DBD12**	**Загрузить значение из двойного слова данных DBD12 в AKKU 1. Вычесть содержимое AKKU 1 из AKKU 2. Запомнить результат в AKKU 1.**
-R		**Копировать содержимое AKKU 3 в AKKU 2. Разделить содержимое AKKU 2 на содержимое AKKU 1. Запомнить результат в**
/R		**AKKU 1 Передать результат (AKKU 1) в двойное слово данных DBD16.**
T	**DBD16**	

10.3. Инкрементирование и декрементирование

Описание

С помощью операций INC (инкрементировать AKKU 1) и DEC (декрементировать AKKU 1) Вы можете выполнять следующие функции:

- Операция INC увеличивает содержимое младшего байта в младшем слове AKKU 1 на 8-битную константу, заданную в команде. Эта константа может лежать в диапазоне от 0 до 255.
- Операция DEC уменьшает содержимое младшего байта в младшем слове AKKU 1 на 8-битную константу, заданную в команде. Эта константа может лежать в диапазоне от 0 до 255.

Операции INC и DEC выполняются независимо от VKE. Эти операции не влияют на VKE, не изменяют ни одного бита в слове состояния.

Указание

Эти операции не годятся для арифметических операций (16 или 32 бита), так как не происходит перенос из младшего байта младшего слова AKKU 1 в старший байт младшего слова AKKU 1. Для арифметических операций (16 или 32 бита) используйте операцию +I или +D.

Пример

Следующий пример программирования показывает, как работает операция INC внутри программного цикла (Loop), который был запущен посредством условного перехода.

AWL			Объяснение
			Тело цикла.
	L	**1**	**Установить счетчик цикла на значение ”1”.**
	T	**MB10**	
M1:	**L**	**MB10**	**Загрузить содержимое меркерного байта MB10 в AKKU 1.**
	INC	**1**	**Увеличить счетчик цикла на ”1”.**
	T	**MB10**	**Передать содержимое AKKU 1 в меркерный байт MB10.**
	.		**Командная часть, которая обрабатывается 5 раз.**
	.		
	L	**B#16#5**	
	<=	**I**	
	SPB	**M1**	**Если программа не выполнила программный цикл пятикратно, то возвратиться на операцию цикла.**

10.4 +AR1 и +AR2: Прибавление константы к адресному регистру 1 или адресному регистру 2

Описание

С помощью операций +AR1 и +AR2 Вы можете прибавить значение к содержимому адресного регистра 1 или адресного регистра 2:

Таблица 10–1. Прибавление значения к содержимому адресных регистров

Операция	Операнд	Функция
+AR1	-	Прибавляет содержимое младшего слова AKKU 1 к содержимому адресного регистра 1.
+AR2	-	Прибавляет содержимое младшего слова AKKU 1 к содержимому адресного регистра 2.
+AR1	P#байт.бит: (Диапазон от 0.0 до 4095.7)[1]	Прибавляет константу указателя к содержимому адресного регистра 1.
+AR2	P#байт.бит: (Диапазон от 0.0 до 4095.7)[1]	Прибавляет константу указателя к содержимому адресного регистра 2.

[1] Биты 24, 25 и 26 адресного регистра не изменяются. Эти биты указывают область памяти.

Указание

Адресный регистр AR2 используется при обработке мультиэкземпляров. Поэтому, если Вы программируете команду ”+AR2”, то Вы должны предварительно “сохранить” содержимое AR2 и впоследствии снова загрузить его обратно.

Примеры

Ниже приведены некоторые примеры с операциями +AR1 и +AR2. Если Вы загружаете значение в формате указателя в AKKU 1, а затем используете одну из операций +AR1 и +AR2, как в первых двух командах в примере, то в Вашем распоряжении имеется диапазон значений от 0.0 до 8191.7.

AWL	Объяснение
L P#250.7	**Загрузить константу указателя (250.7) в AKKU 1.**
+AR1	**Прибавить содержимое аккумулятора (250.7) к содержимому адресного регистра 1.**
	Из-за мультиэкземпляров, которые используют AR2 как базу
T AR2 #SAVE_AR2	
	Прибавить константу указателя (126.7) к содержимому адресного регистра
+AR2 P#126.7	
.	**Восстановить AR2.**
.	
L AR2 #SAVE_AR2	