ПРОГРАММНЫЙ СИНТЕЗ

Постановка задачи

Требуется на языке ассемблера TMS320C6x разработать программу, обеспечивающую решение фрагмента некоторой задачи, заданного в табл. 1 в соответствии с номером варианта. Номер варианта M определяется по номеру студента в списке группы.

– для вариантов, помеченных звездочкой, двоичные слова блока, размещенные в процессорной памяти, начиная с ячейки m_2m_1+1;

– для остальных вариантов за исключением 13 – блок из 8 2-разрядных двоичных слов, размещенный в ячейке m_2m_1 процессорной памяти.

m2m1 – 2 последние цифры номера зачетной книжки

Структура блока произвольна и заранее не известна.

Таблица 1. Содержание фрагмента задачи

M	Фрагмент задачи
0	*Установить значение наибольшего по величине двоичного слова в блоке.
1	Определить количество наибольших по величине двоичных слов в блоке, ели эта величина известна и хранится в ячейке памяти m_2m_1+16.
2	*Определить порядковые номера, занимаемые наибольшими по величине двоичными словами в блоке, ели эта величина известна и хранится в ячейке памяти m_2m_1+16. Результаты сохранить в памяти, начиная с ячейки $1m_2m_1$.
3	*Сформировать новый блок без наименьших по величине двоичных слов исходного блока, ели эта величина известна и хранится в ячейке памяти m_2m_1+16.
4	Сформировать новый блок без наименьших по величине двоичных слов исходного блока, ели их порядковые номера хранятся в ячейках памяти с $1m_2m_1–2$ по $1m_2m_1+2$.
5	*Сформировать новый блок из нечетных по величине двоичных слов исходного блока.
6	Сформировать новый блок, где двоичные слова нечетных позиций соответствуют исходному блоку, а четных позиций представляют собой результат поразрядной инверсии двоичных слов исходного блока.
7	*Сформировать новый блок, двоичные слова которого представляют собой дополнительный код соответствующих двоичных слов исходного блока.
8	*Определить количество единичных бит в блоке.
9	Определить количество нулевых бит в блоке.
10	Сформировать новый блок из двоичных слов, имеющих точно одну единицу.
11	Выполнить поразрядную дизъюнкцию над двоичными словами со следующими порядковыми номерами: 1 и 8, 2 и 7, 3 и 6, 4 и 5. Результаты сохранить в памяти, начиная с ячейки $1m_2m_1$.
12	*Для каждого двоичного слова выполнить операцию $y_0 \oplus y_1$. Результаты сохранить в памяти, начиная с ячейки $1m_2m_1$.
13	В последовательности из 8 блоков в памяти, начиная с ячейки $1m_2m_1$, сохраняется только блок, число нулевых старших разрядов которого меньше 5. При этом сохранению подлежат лишь 8 разрядов, начиная с первой значащей единицы.
14	*Сформировать новый блок, состоящий только из старших разрядов двоичных слов.
В остальных вариантах сформировать новый блок с измененным порядком следования двоичных слов (ДС) исходного блока:	
15	*$ДС_1$ $ДС_8$ $ДС_2$ $ДС_7$ $ДС_3$ $ДС_6$ $ДС_4$ $ДС_5$
16	*$ДС_1$ $ДС_3$ $ДС_5$ $ДС_7$ $ДС_2$ $ДС_4$ $ДС_6$ $ДС_8$
17	*$ДС_1$ $ДС_5$ $ДС_2$ $ДС_6$ $ДС_3$ $ДС_7$ $ДС_4$ $ДС_8$
18	*$ДС_4$ $ДС_5$ $ДС_3$ $ДС_6$ $ДС_2$ $ДС_7$ $ДС_1$ $ДС_8$
19	*$ДС_2$ $ДС_4$ $ДС_6$ $ДС_8$ $ДС_1$ $ДС_3$ $ДС_5$ $ДС_7$
20	*$ДС_2$ $ДС_4$ $ДС_1$ $ДС_3$ $ДС_6$ $ДС_8$ $ДС_5$ $ДС_7$
21	*$ДС_1$ $ДС_2$ $ДС_5$ $ДС_6$ $ДС_3$ $ДС_4$ $ДС_7$ $ДС_8$
22	*$ДС_7$ $ДС_5$ $ДС_3$ $ДС_1$ $ДС_8$ $ДС_6$ $ДС_4$ $ДС_2$

23	*ДС$_4$ ДС$_3$ ДС$_2$ ДС$_1$ ДС$_8$ ДС$_7$ ДС$_6$ ДС$_5$
24	*ДС$_8$ ДС$_7$ ДС$_6$ ДС$_5$ ДС$_4$ ДС$_3$ ДС$_2$ ДС$_1$
25	*ДС$_2$ ДС$_1$ ДС$_4$ ДС$_3$ ДС$_6$ ДС$_5$ ДС$_8$ ДС$_7$
26	*ДС$_3$ ДС$_6$ ДС$_1$ ДС$_4$ ДС$_7$ ДС$_2$ ДС$_5$ ДС$_8$
27	*ДС$_6$ ДС$_3$ ДС$_8$ ДС$_5$ ДС$_2$ ДС$_7$ ДС$_4$ ДС$_1$
28	*ДС$_1$ ДС$_3$ ДС$_7$ ДС$_5$ ДС$_2$ ДС$_4$ ДС$_8$ ДС$_6$
29	ДС$_8$ ДС$_7$ ДС$_6$ ДС$_5$ ДС$_4$ ДС$_3$ ДС$_2$ ДС$_1$

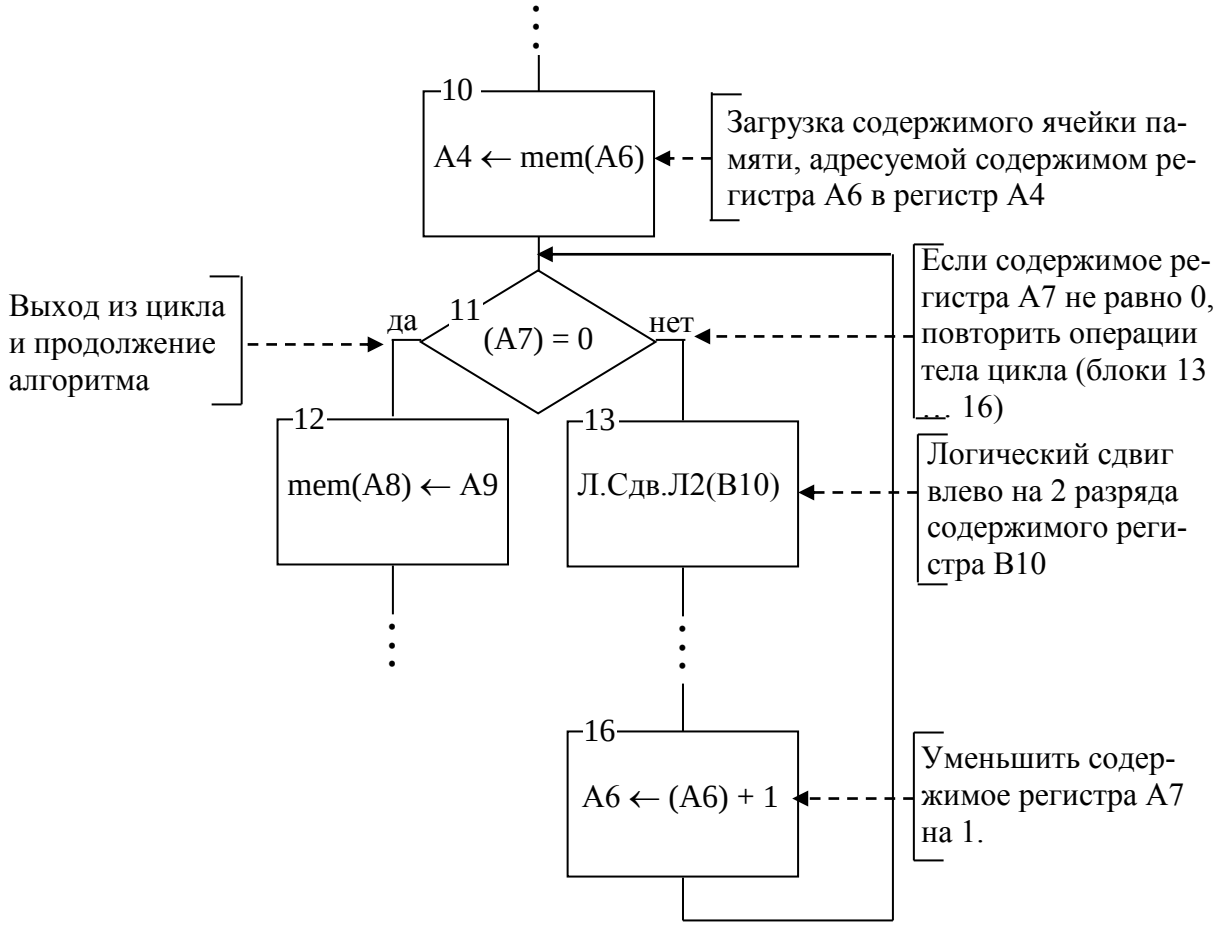


Рис. 1. Фрагмент блок-схемы алгоритма

Для разработанной программы определить число пакетов выборки и выполняемых пакетов.

Решение поставленной задачи

В данной работе достаточно ограничиться двумя этапами программного синтеза: разработкой детального алгоритма, отражающего идею решения задачи, и его описанием на языке ассемблера TMS320C6x.

2.2.1. Требования к построению детального алгоритма

Обычно алгоритм представляется блок-схемой. В каждом блоке в словесной или символической форме описывается операция, выполняемая на данном шаге решения задачи. Алгоритмические блоки, за исключением блоков «начало» и «конец» нумеруются в порядке их следования сверху вниз и слева направо. Для примера на рис. 1 приведена блок-схема, характерная для циклических (многократное повторение одних и тех же вычислений, но для различных исходных данных) процессов. В примере операции описываются в символической форме.

Требования к написанию программы

Программа представляет собой последовательное описание алгоритмических блоков на языке программирования, и сопровождаться комментариями.

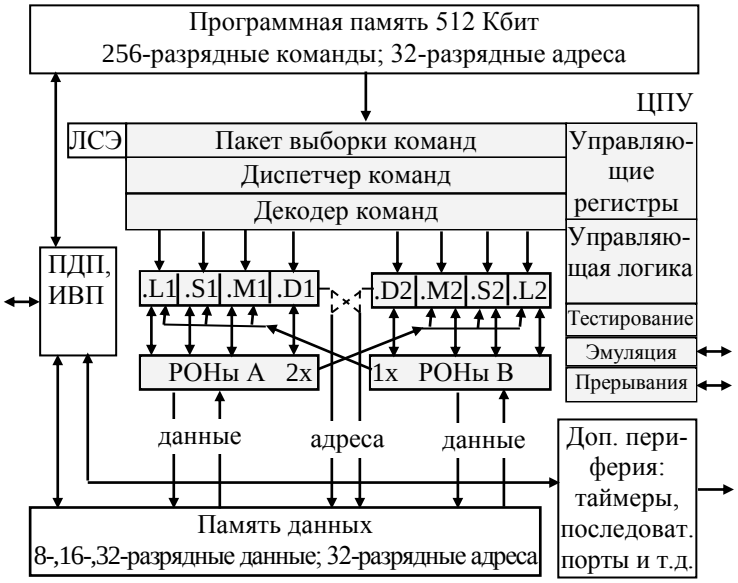
При написании программы на языке ассемблера необходимо учитывать архитектурные особенности процессора. Применительно к TMS320C6x основными из таких особенностей являются:

- циклические вычисления реализуются только с помощью команды перехода;
- условной может быть любая команда;
- параллельное выполнение до 8 команд.

Структурно ассемблерная программа состоит из последовательности строк, в каждой из которых размещается только одна команда. Командные строки не нумеруются.

ОСНОВНЫЕ СВЕДЕНИЯ О ПРОЦЕССОРЕ TMS320C6x

1. Структура



ЛСЭ – логика снижения энергопотребления.

ИВП – интерфейс внешней памяти.

ПДП – контроллер прямого доступа в память.

1x и 2x – перекрестные линии.

.L, .S, .M, .D – функциональные устройства (модули), из которых .L, .S и .D – АЛУ, .M – умножитель.

2. Основные команды для чисел с фиксированной запятой

Условности при описании команд:

R1 – регистр 1-го операнда, R2 – регистр 2-го операнда, Rsm – регистр смещения, Rbas – регистр базы, Rres – регистр результата, cst – константа.

Тип операнда обозначается аббревиатурой: int – 32-разрядное целое, short – короткое (16-разрядное), long – длинное (40-разрядное).

Перед каждой аббревиатурой типа буква s означает величину со знаком (т.е. старший разряд является знаковым, а остальные разряды представляют число в дополнительном коде), а буква u – величину без знака (т.е. все разряды представляют число в прямом коде).

Любой тип, начинающийся буквой x, означает, что операнд может исходить из противоположного регистрового файла.

Слот задержки – интервал времени (число тактов) между началом выполнения команды и моментом времени, когда результат становится доступным для чтения.

ZERO Rres Обнуление регистра

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2, .D1 или .D2	нет	нет	sint	1-тактная	0
.L1 или .L2	нет	нет	slong		

ABS R1, Rres Абсолютная величина целого

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	xsint	нет	sint	1-тактная	0

Описание: абсолютная величина R1 устанавливается в Rres.

ADD R1, R2, Rres Сложение знаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint	xsint	sint	1-тактная	0
	sint	xsint	slong		
	xsint	slong	slong		
	scst5	xsint	sint		
	scst5	slong	slong		
.S1 или .S2	sint	xsint	sint		
	scst5	xsint	sint		
.D1 или .D2	sint	sint	sint		
	ucst5	sint	sint		

Описание: R2 добавляется к R1 (для устройств .L и .S) или R1 добавляется к R2 (для устройств .D) и результат устанавливается в Rres.

ADDU R1, R2, Rres Сложение беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	uint	xuint	ulong	1-тактная	0
	xuint	ulong	ulong		

Описание: R2 добавляется к R1 и результат устанавливается в Rres.

ADDK cst, Rres Сложение целых с использованием знаковой 16-разрядной константы

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	scst16	Rres	uint	1-тактная	0

Описание: 16-разрядная знаковая константа добавляется к регистру Rres и результат устанавливается в Rres.

ADD2 R1, R2, Rres Два 16-разрядных целых добавляются к старшей и младшей половинам регистра

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	sint	xsint	sint	1-тактная	0

Описание: старшая и младшая половины R1 добавляются к старшей и младшей половинам R2, соответственно. Перенос от младшей половины результата к старшей не производится. Результат устанавливается в Rres.

AND R1, R2, Rres Поразрядное И

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2, .L1 или .L2	uint	xuint	uint	1-тактная	0
	scst5	xuint	uint		

Описание: над R1 и R2 выполняется поразрядное И. Результат устанавливается в Rres. Константа распределяется на 32-х разрядах.

B R1 Переход

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	метка	нет	переход	Ветвление	5
.S2	xuint	нет			

CLR R1, R2, R3, Rres Очистить область бит беззнакового целого

Устройство	Тип операндов			Тип результата	Тип команды	Такты задержки
	R1	R2	R3			
.S1 или .S2	uint xuint	ucst15 uint	ucst15 нет	uint uint	1-тактная	0

Описание: область в R1, заданная константами R2 и R3, сбрасывается в ноль, и результат устанавливается в Rres. Константа R2 определяет номер младшего бита (относительно нулевого разряда R1), а константа R3 – номер старшего бита (относительно нулевого разряда R1) области, которая должна очищаться. Константы могут быть определены десятью младшими битами регистра R2, где первая константа является битами 0-4, а вторая – битами 5-9.

CMPEQ R1, R2, Rres Сравнение знаковых целых на равенство

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint scst5 xsint scst5	xsint xsint slong slong	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если R1 = R2, то в Rres записывается 1. В противном случае в Rres записывается 0.

CMPGT R1, R2, Rres Сравнение знаковых целых в отношении "больше"

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint scst5 xsint scst5	xsint xsint slong slong	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если R1 > R2, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPGTU R1, R2, Rres Сравнение беззнаковых целых в отношении "больше"

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	uint ucst4 xuint ucst4	xuint xuint ulong ulong	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если R1 > R2, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPLT R1, R2, Rres Сравнение знаковых целых в отношении "меньше"

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint scst5 xsint scst5	xsint xsint slong slong	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если R1 < R2, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPLTU R1, R2, Rres Сравнение беззнаковых целых в отношении "меньше"

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	uint ucst4 xuint ucst4	xuint xuint ulong ulong	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если R1 < R2, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

EXT R1, R2, R3, Rres Выделение и расширение знаком области бит

Устройство	Тип операндов			Тип результата	Тип команды	Такты задержки
	R1	R2	R3			
.S1 или .S2	sint xsint	ucst5 uint	ucst5 нет	sint sint	1-тактная	0

Описание: область в R1, определённая константами R2 и R3, извлекается и размещается в Rres, расширяясь знаком до 32-х разрядов. Номер старшего бита области равен разности (31 – первая константа(R2)), а номер младшего бита области – разности (вторая константа(R3) - первая константа(R2)). Выделенная область переносится в R1, сдвигается влево на число разрядов, равное первой константе, а затем – вправо на число разрядов, равное второй константе. При сдвигах вправо освобождавшиеся разряды заполняются знаком (старшим разрядом области). Константы могут быть определены как 10 младших бит регистра R2, где первая константа является битами 5-9, а вторая – битами 0-4.

EXTU R1, R2, R3, Rres Выделение и расширение нулями области бит

Устройство	Тип операндов			Тип результата	Тип команды	Такты задержки
	R1	R2	R3			
.S1 или .S2	uint xuint	ucst5 uint	ucst5 нет	uint uint	1-тактная	0

Описание: всё также как и в команде EXT, но заполнение производится нулём.

LDB, LDBU, LDH, LDHU, LDW R1, Rres Загрузка из памяти с 5-разрядной беззнаковой константой смещения или с регистром смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.D1 или .D2	См. синтаксис R1	нет	Содержимое ячейки памяти	Загрузка	4

Описание: каждая из этих команд загружает из памяти в регистр Rres РОН. Синтаксис R1, определяющий правило формирования адреса:

Синтаксис	Выполняемая модификация регистра Rbas
*+Rbas[Rsm] или *+Rbas[ucst5]	*Положительное смещение*. К Rbas добавляется Rsm или ucst5, причём Rbas не изменяется. Адресом является результат суммирования.
*-Rbas[Rsm] или *-Rbas[ucst5]	*Отрицательное смещение*. Из Rbas вычитается Rsm или ucst5, причём Rbas не изменяется. Адресом является результат вычитания.
*++Rbas[Rsm] или *++Rbas[ucst5]	*Преинкремент*. К Rbas добавляется Rsm или ucst5, изменяя Rbas. Адресом является изменённое содержимое Rbas.
*- -Rbas[Rsm] или *- -Rbas[ucst5]	*Предекремент*. Из Rbas вычитается Rsm или ucst5, изменяя Rbas. Адресом является изменённое содержимое Rbas.
*Rbas++ [Rsm] или *Rbas++ [ucst5]	*Постинкремент*. К Rbas добавляется Rsm или ucst5, изменяя Rbas. Адресом является содержимое Rbas до его изменения.
*Rbas- -[Rsm] или *Rbas- -[ucst5]	*Постдекремент*. Из Rbas вычитается Rsm или ucst5, изменяя Rbas. Адресом является содержимое Rbas до его изменения.

Если смещение (Rsm или ucst5) не задаётся, ассемблер назначает нулевое смещение, а инкременты и декременты равны 1. Rbas и Rsm должны быть в том же регистровом файле и на той же стороне, что и используемое устройство .D. Rsm и cst5 до операции по формированию адреса сдвигаются влево на 0 (LDB и LDBU), 1 (LDH и LDHU) или 2 (LDW) разряда. Адресная арифметика (сложение и вычитание) по умолчанию выполняется в линейном способе. Однако, для A4-A7 и B4-B7 может быть изменён на циклический способ путём записи соответствующей величины в регистр AMR. Для команд LDB(U) и LDH(U) загружаются только младшие 8 и 16 бит, соответственно, при этом остальные разряды заполняются знаком (LDB и LDH) или нулями (LDBU и LDHU). Для LDW заполняются все 32 разряда Rres.

LDB, LDBU, LDH, LDHU, LDW R1, Rres Загрузка из памяти с 15-разрядной беззнаковой константой смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	Синтаксис R1	R2			
.D2	*+B14[ucst15] или *+B15[ucst15]	нет	Содержимое ячейки памяти	Загрузка	4

Описание: каждая из этих команд загружает из памяти в регистр Rres РОН. До операции по формированию адреса (вычитание не поддерживается) cst15 сдвигается влево на 0 (LDB и LDBU), 1 (LDH и LDHU) или 2 (LDW) разряда. Адресная арифметика всегда выполняется в линейном способе. Для команд LDB(U) и LDH(U) загружаются только младшие 8 и 16 бит, соответственно, при этом остальные разряды заполняются знаком (LDB и LDH) или нулями (LDBU и LDHU). Для LDW заполняются все 32 разряда Rres.

LMBD R1, R2, Rres Обнаружение самого левого бита

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	uint cst5	xuint xuint	uint uint	1-тактная	0

Описание: младший бит R1 определяет, что искать в R2– самую левую единицу или самый левый ноль. Информация о количестве бит слева от первой 1 или первого 0 при поиске 1 или 0, соответственно, размещает-

ся в Rres.

MPY, MPYU, MPYUS, MPYSU R1, R2, Rres Умножение 16 x 16 младших бит знаковых или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	sмл.16бит	xsмл.16бит	sint	Умножение 16 на 16	1
	uмл.16бит	xuмл.16бит	uint		
	uмл.16бит	xsмл.16бит	sint		
	sмл.16бит	xuмл.16бит	sint		
	scst5	xsмл.16бит	sint		
	scst5	xuмл.16бит	sint		

Описание: 16 младших бит R1 умножаются на 16 младших бит R2 и результат размещается в Rres. В команде MPY оба операнда знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYH, MPYHU, MPYHUS, MPYHSU R1, R2, Rres Умножение 16 x 16 старших бит знаковых или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	sct.16бит	xsct.16бит	sint	Умножение 16 на 16	1
	uct.16бит	xuct.16бит	uint		
	uct.16бит	xsct.16бит	sint		
	sct.16бит	xuct.16бит	sint		

Описание: 16 старших бит R1 умножаются на 16 старших бит R2 и результат размещается в Rres. В команде MPY оба операнда знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYHL, MPYHLU, MPYHULS, MPYHSLU R1, R2, Rres Умножение 16 старших бит на 16 младших бит знаковых

или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	sct.16бит	xsмл.16бит	sint	Умножение 16 на 16	1
	uct.16бит	xuмл.16бит	uint		
	uct.16бит	xsмл.16бит	sint		
	sct.16бит	xuмл.16бит	sint		

Описание: 16 старших бит R1 умножаются на 16 младших бит R2 и результат размещается в Rres. В команде MPYHL оба операнда знаковые, а в команде MPYHLU – оба без знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYLH, MPYLHU, MPYLUHS, MPYLSHU R1, R2, Rres Умножение 16 младших бит на 16 старших бит знаковых

или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	sмл.16бит	xsct.16бит	sint	Умножение 16 на 16	1
	uмл.16бит	xuct.16бит	uint		
	uмл.16бит	xsct.16бит	sint		
	sмл.16бит	xuct.16бит	sint		

Описание: 16 младших бит R1 умножаются на 16 старших бит R2 и результат размещается в Rres. В команде MPYHL оба операнда знаковые, а в команде MPYHLU – оба без знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MV R1, Rres Перемещение из одного регистра в другой

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	xsint	нет	sint	1-тактная	0
.D1 или .D2	sint	нет	sint		
.L1 или .L2	slong	нет	slong		

MVK R1, Rres Перемещение 16-разрядной константы в пределах регистра и расширение знаком

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	scst16	нет	sint	1-тактная	0

Описание: 16-разрядная константа является расширенной знаком (т.е. свободные старшие разряды Rres заполняются знаком константы) и размещается в Rres.

MVKH, MVKLH R1, Rres Перемещение 16-разрядной константы в старшие разряды регистра

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	scst16	нет	sint	1-тактная	0

Описание: 16 старших (MVKH) или 16 младших (MVKLH) разрядов константы загружаются в старшие разряды Rres. 16 младших разрядов Rres остаются неизменными.

NOP R1 Нет операции

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
нет	ucst4	нет	нет	NOP	0

Описание: R1 задаёт число тактов, в течение которых никакие операции (за исключением перехода) не производятся. NOP без операнда рассматривается как NOP 1.

NOT R1, Rres Поразрядное НЕ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	xuint	нет	uint	1-тактная	0

OR R1, R2, Rres Поразрядное ИЛИ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	uint scst5	xuint xuint	uint uint	1-тактная	0

Примечание: константа представляется в дополнительном коде и расширяется знаком вплоть до 32-х бит.

SET R1, R2, R3, Rrest Установка области бит

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	uint xuint	ucst5 uint	ucst5 нет	uint uint	1-тактная	0

Описание: область в R1, определенная константой R2 и константой R3, размещается в Rrest, после чего устанавливается во все единицы. Первая (R2) и вторая (R3) константы определяют номер (относительно нулевого разряда), соответственно, младшего и старшего разряда области, которая должна устанавливаться во все единицы. Константы могут быть определены как десять младших бит регистра R2, причём биты 0-4 соответствуют второй константе, а биты 5-9 – первой.

SHL R1, R2, Rrest Арифметический сдвиг влево

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	xsint slong xuint xsint xuint slong	uint uint uint ucst5 ucst5 ucst5	sint slong ulong sint ulong slong	1-тактная	0

Описание: операнд R1 загружается в Rrest, после чего перемещается влево на число разрядов, определённых константой R2. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если 39 < R2 < 64, R1 перемещается влево на 40 разрядов. Освобождающиеся при сдвиге разряды заполняются нулями.

SHR R1, R2, Rrest Арифметический сдвиг вправо

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			

.S1 или .S2	xsint slong xsint slong	uint uint ucst5 ucst5	sint slong sint slong	1-тактная	0

Описание: операнд R1 загружается в Rrest, после чего перемещается вправо на число разрядов, определённых константой R2. Результат расширяется знаком. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если 39 < R2 < 64, R1 перемещается вправо на 40 разрядов.

SHRU R1, R2, Rrest Логический сдвиг вправо

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	xuint ulong xuint ulong	uint uint ucst5 ucst5	uint ulong uint ulong	1-тактная	0

Описание: операнд R1 загружается в Rrest, после чего перемещается вправо на число разрядов, определённых константой R2. Результат расширяется нулями. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если 39 < R2 < 64, R1 перемещается вправо на 40 разрядов.

STB, STH, STW R*R1 Загрузка в память с 5-разрядной беззнаковой константой смещения или с регистром смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.D1 или .D2	См. синтаксис R1	нет	нет	Хранение	0

Описание: каждая из этих команд загружает в память из регистра R РОН. Синтаксис R1 и выполнение Rbas и Rsm такие же, как у аналогичных команд LD.

STB, STH, STW R*R1 Загрузка в память с 15-разрядной беззнаковой константой смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.D1 или .D2	См. синтаксис R1	нет	нет	Хранение	0

Описание: каждая из этих команд загружает в память из регистра R РОН. Синтаксис R1 такой же, как у аналогичных команд LD.

SUB, SUBU R1, R2, Rres Вычитание знакового или беззнакового целого без насыщения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint xsint sint xsint uint xuint scst5	xsint sint xsint sint xuint uint slong	sint sint slong slong ulong ulong slong	1-тактная	0
.S1 или .S2	sint scst5	xsint xsint	sint sint		
.D1 или .D2	sint sint	sint ucst5	sint sint		

Описание: R2 вычитается из R1 и результат размещается в Rres.

XOR R1, R2, Rres Исключающее ИЛИ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	uint scst5	xuint xuint	uint uint	1-тактная	0

Описание: поразрядное исключающее ИЛИ выполняется между R1 и R2. Результат устанавливается в Rres. Константа расширена знаком до 32-х бит.