ЛАБОРАТОРНАЯ РАБОТА № 3

РАБОТА С ПАМЯТЬЮ. ПОДПРОГРАММЫ И ЦИКЛЫ

Цель работы: изучение структуры и функций блока микропрограммного управления, микропрограммирование обработки данных, записанных в оперативной памяти (ОП), с использованием циклов и подпрограмм.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. по материалам раздела 2.4 лабораторной работы № 1 и описания инструкций УПМ (поле CHA) ознакомиться с особенностями работы блока микропрограммного управления;
2. по материалам раздела 2 ознакомиться с особенностями работы с оперативной памятью;
3. по материалам раздела 3 ознакомиться с принципами микропрограммирования подпрограмм и циклов;
4. используя как образец микропрограмму, приведенную в разделе 4, разработать, закодировать и ввести микропрограмму решения задачи;
5. ввести исходные данные, отладить микропрограмму в режиме МИКРОКОМАНДА, фиксируя изменения состояния модели после выполнения каждой микрокоманды;
6. выполнить микропрограмму на различных наборах исходных данных в режиме АВТОМАТ.

2. РАБОТА С ОПЕРАТИВНОЙ ПАМЯТЬЮ

Ввод данных в ОП производится, как было описано в работе №1. Возможны просмотр и корректировка данных в приостановках после выполнения очередного шага моделирования.

Адресация ОП выполняется различными способами, которые рассматриваются ниже. При этом 16-ричный адрес не должен превышать 3FF.

Прямая адресация. Адрес слова ОП задается в поле CONST микрокоманды (МК). Загрузка адреса происходит через процессор по цепочке «шина DA - MR - ALU - SDA – ARAM» и требует отдельной МК с кодом 2 в поле WM и кодом 5 в поле SRC.

Косвенная регистровая адресация. Адрес хранится в одном из регистров РЗУ, куда он может быть загружен предварительно из поля CONST или сформирован иным образом. Адрес регистра задается полем B. Содержимое регистра передается в ARAM из RGB, для чего в поле WM нужно записать 3. Параллельно в АЛУ могут выполняться различные преобразования информации.

Автоинкрементная адресация. Адрес также передается в ARAM через RGB. Одновременно к регистру прибавляется 2. Это можно сделать,

записав число 2 в поле CONST и указав CONST в качестве источника операнда R.

Автодекрементная адресация. Из регистра вычитается 2, полученное значение передается в ARAM с выхода SDA.

Косвенная адресация. Адрес ОП выбирается из другой ячейки ОП, адресуемой полем CONST или регистром РЗУ. Передача адреса в ARAM после его чтения из ОП производится через операционный блок по цепочке «RGR - MS - ALU - SDA – ARAM».

3. ПОДПРОГРАММЫ И ЦИКЛЫ

Работа с подпрограммами выполняется с помощью функций CJS (условный вызов) и CRTN (условный возврат) схемы УПМ. Чтобы сделать их безусловными, в поле JFI записывается 4 (бит J=1). При вызове адрес возврата запоминается в стеке и происходит переход по адресу подпрограммы, заданному в поле CONST. При возврате происходят переход по адресу из стека и декремент указателя стека STP.

Для организации циклов с числом повторений N в счетчик RACT загружается значение N-1 из поля CONST с помощью функции LDCT. Для возврата в начало цикла используют функцию RPCT. Если RACT=0, следующая МК выбирается в естественном порядке. В противном случае происходят декремент счетчика и переход по адресу из поля CONST.

4. МИКРОПРОГРАММИРОВАНИЕ АЛГОРИТМОВ

Пример. Задан массив ARR1 из M чисел, который записан в ОП начиная с адреса NADDR. Подсчитать сколько единиц в двоичном представлении в каждом числе массива и результат записать в другой массив ARR2, начиняя с адреса KADDR. Подсчет количества единиц в двоичном представлении числа оформить как подпрограмму.

Алгоритм подсчета количества единиц в двоичном представлении числа был рассмотрен в лабораторной работе № 2 (рис. 3 б). На рис. 3 а представлен алгоритм работы с массивами. Для адресации памяти используем автоинкрементную адресацию. Для этого в регистр SI загружаем начальный адрес массива NADDR, а для обращения к следующей ячейке памяти осуществляем инкремент SI (SI:=SI+2). Аналогично, в регистр DI загружаем начальный адрес массива KADDR, а для обращения к следующей ячейке памяти осуществляем инкремент DI (DI:=DI+2). Запись регистра в квадратных скобках означает, что мы работаем не с содержимым регистра, а с содержимым ячейки ОП, адрес которой находится в заданном регистре. Исходное число загружается из массива ARR1 в регистр AX. В подпрограмме подсчитывается количество единиц в двоичном представлении числа, и результат помещается в DX и далее в массив ARR2 по вычисленному адресу в ОП.

При интерпретации программы примем, что основная микропрограмма (табл. 14) начинается с адреса 00h, подпрограмма (табл.

15) - с адреса 10h, исходный массив из десяти чисел записан, начиная с адреса 100h, результирующий массив начинается с адреса 150h, число повторений циклов заносится в счетчик RACT.

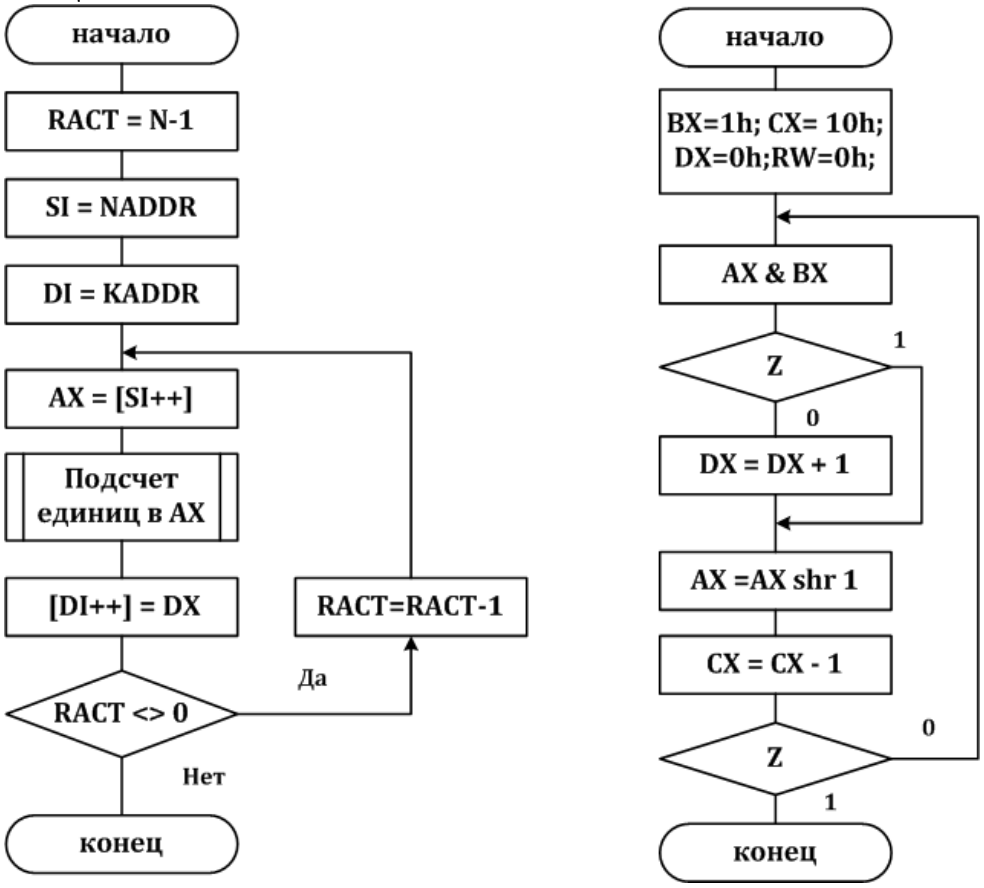

Рис. 3. Схемы алгоритмов:
а – Алгоритм работы с массивами;
б – Алгоритм подсчета количества единиц в двоичном представлении числа

Таблица 14. Микропрограммы работы с массивами

Адрес МК	Операция	Поле	Значение	Функция
00	RACT = 9	CHA CONST	6 0009	LDCT Счетчик элементов массива
01	SI = NADDR	B SRC DST CONST	6 5 4 0100h	SI CONST, RGB РЗУ[B]=SDA Нач. адрес ARR1
02	DI = KADDR	B SRC DST CONST	7 5 4 0150h	DI CONST, RGB РЗУ[B]=SDA Нач. Адрес ARR2

03	M: ARAM = SI SI = SI +2	B WM SRC ALU DST CONST	6 3 5 3 4 0002	SI ARAM = RGB CONST, RGB R + S РЗУ[B]=SDA инкремент
04	AX = [ARAM]	B MEM DST	0 5 1	AX Чтение слова из ОП РЗУ[B]:= RGR
05	Вызов подпрограммы	JFI CHA CONST	4 1 10	Б/у переход CJS Адрес подпрограммы
06	ARAM = DI SI = DI +2	B WM SRC ALU DST CONST	7 3 5 3 4 0002	DI ARAM = RGB CONST, RGB R + S РЗУ[B]=SDA инкремент
07	[ARAM] = DX if RACT<>0 then RACT:= RACT-1 goto M	A WM MEM CHA CONST	2 1 7 4 0003	DX RGW = SDA Запись слова в ОП RPCT Адрес метки M
08	STOP	JFI	5	Останов

Таблица 15. Микропрограмма для подсчета единиц в числе

Адрес МК	Операция	Поле	Значение	Функция
10	BX = 0001h	B SRC DST CONST	3 5 4 0001	BX CONST, RGB РЗУ[B]=SDA Маска
11	CX = 0010h	B SRC DST CONST	1 5 4 0010	CX CONST, RGB РЗУ[B]=SDA Счетчик циклов
12	DX = 0000h	B SRC DST CONST	2 5 4 0000	DX CONST, RGB РЗУ[B]=SDA Счетчик единиц
13	RW = 0000h	B SRC DST	F 5 4	RW CONST, RGB РЗУ[B]=SDA

		CONST	0000	Вспомогательный регистр обнуляется
14	M1: AX & BX if Z then goto M2	A B ALU CC CHA CONST	0 3 9 1 3 0016	AX BX R & S JZ CJP Адрес перехода
15	DX = DX + 1	A B CCX DST	2 2 1 4	DX DX C0=1 РЗУ[B]=SDA
16	M2: AX = AX shr 1	SH N DST	2 1 4	ЛС сдвиг вправо Сдвиг на 1 разряд РЗУ[B]=SDA
17	CX = CX-1 if not Z then goto M1	A B ALU DST CC JFI CHA CONST	F 1 1 4 1 1 3 0014	RW CX S – R – 1 + C0 РЗУ[B]=SDA JNZ I=1 CJP Адрес перехода
18	Безусловный возврат из подпрограммы	JFI CHA	4 5	Б/у переход CRTN

5. СОДЕРЖАНИЕ ОТЧЕТА

В отчет входят следующие пункты:

1. схемы алгоритмы решаемой задачи;
2. микропрограмма и микроподпрограмма с подробными комментариями;
3. исходные данные с указанием их размещения в ОП;
4. результаты решения задачи в режиме МИКРОКОМАНДА (трасса) и АВТОМАТ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1) Опишите структуру блока микропрограммного управления и функции его составляющих. Для каких целей используются стек, счетчик RACT, мультиплексор кода условия в вашей микропрограмме?

2) Какие способы формирования адреса следующей МК реализуются в БМУ?

3) Проанализируйте формирование условий перехода в зависимости от значения поля СС. В каких случаях используется инверсия условия? Когда следует использовать дополнительный регистр флажков?

4) Какие инструкции БМУ используются для организации циклов и подпрограмм? Можно ли в вашей микропрограмме организовать циклы и подпрограммы иначе, чем вы это сделали? Как это повлияет на качество микропрограммы?

5) Опишите способы адресации ОП, используемые в процессоре. Какие из них имеют место в вашей микропрограмме?

7. ЗАДАНИЯ НА МИКРОПРОГРАММИРОВАНИЕ

1. Заданы два массива положительных чисел A (байты) и B (байты). Сформировать массив C (слова), каждый элемент которого есть произведение A[I]*B[I]. Умножение двух чисел по алгоритму с анализом старшего бита множителя со сдвигом множимого оформить как подпрограмму.

2. Заданы два массива положительных чисел A (байты) и B (байты). Сформировать массив C (слова), каждый элемент которого есть произведение A[I]*B[I]. Умножение двух чисел по алгоритму с анализом младшего бита множителя со сдвигом множимого оформить как подпрограмму.

3. Заданы два массива положительных чисел A (слова) и B (байты). Сформировать массив C (байты), каждый элемент которого есть частное A[I] / B[I]. Деление двух чисел по алгоритму с восстановлением остатка оформить как подпрограмму.

4. Заданы два массива положительных чисел A (байты) и B (байты). Сформировать массив C (слова), каждый элемент которого есть произведение A[I]*B[I]. Умножение двух чисел по алгоритму с анализом старшего бита множителя со сдвигом СЧП оформить как подпрограмму.

5. Заданы два массива положительных чисел A (слова) и B (слова). Сформировать массив C (двойные слова), каждый элемент которого есть произведение A[I]*B[I]. Умножение двух чисел по алгоритму с анализом младшего бита множителя со сдвигом СЧП оформить как подпрограмму.

6. Построить массив Q квадратов элементов массива A. Вычисление квадрата оформить как подпрограмму. При вычислении квадрата очередного числа использовать формулу $(X+1)^2=X^2+2X+1$.

7. Задан массив 4-разрядных положительных десятичных чисел. Преобразовать его в массив двоичных чисел. Преобразование выполнить

по формуле $B=((D_1*10+D_2)*10+D_3)*10+D_4$, где D_i - тетрады числа, начиная со старшей, и оформить как подпрограмму.

Рекомендации по микропрограммированию:

– чтобы выделить тетраду, десятичное число необходимо поместить в RGQ, используя функцию сдвигателя ALU=>RGQ. А затем выполняем двойной сдвиг ALU и RGQ влево на 4 бита с установкой SRC=0;

– умножение RgBin на 10 сводится к сложению и сдвигам по формуле: (RgBin * 4 + RgBin) * 2.

8. Сформировать массив остатков от деления элементов исходного массива A на 15. Вычисление остатка оформить как подпрограмму.

Рекомендации по микропрограммированию. Остаток от деления числа на 15 равен остатку от деления на 15 суммы его 16-ричных цифр.

Чтобы выделить тетраду, число необходимо поместить в RGQ, используя функцию сдвигателя ALU=>RGQ. А затем выполняем двойной сдвиг ALU и RGQ влево на 4 бита с установкой SRC=0. Второй способ с помощью маски и сдвигов.

9. Подсчитать число элементов массива A, которые дают ненулевые остатки при делении на 17. Вычисление остатка оформить как подпрограмму.

Рекомендации по микропрограммированию. Остаток от деления на 17 равен остатку от деления на 17 суммы цифр числа, взятых попеременно со знаками плюс и минус, начиная с младшей цифры.

Чтобы выделить тетраду, число необходимо поместить в RGQ, используя функцию сдвигателя ALU=>RGQ. А затем выполняем двойной сдвиг ALU и RGQ влево на 4 бита с установкой SRC=0. Второй способ с помощью маски и сдвигов.

10. Подсчитать число элементов массива A, которые дают ненулевые остатки при делении на 9. Вычисление остатка оформить как подпрограмму.

Рекомендации по микропрограммированию. Остаток от деления числа на 9 равен остатку от деления на 9 суммы его десятичных цифр.

Чтобы выделить тетраду, число необходимо поместить в RGQ, используя функцию сдвигателя ALU=>RGQ. А затем выполняем двойной сдвиг ALU и RGQ влево на 4 бита с установкой SRC=0. Второй способ с помощью маски и сдвигов.

11. Сформировать массив остатков от деления элементов исходного массива A на 11. Вычисление остатка оформить как подпрограмму.

Рекомендации по микропрограммированию. Остаток от деления на 11 равен остатку от деления на 11 суммы цифр числа, взятых попеременно со знаками плюс и минус, начиная с младшей цифры.

Чтобы выделить тетраду, число необходимо поместить в RGQ, используя функцию сдвигателя ALU=>RGQ. А затем выполняем двойной сдвиг ALU и RGQ влево на 4 бита с установкой SRC=0. Второй способ с помощью маски и сдвигов.

12. Заданы два массива положительных чисел A (слова) и B (байты). Сформировать массив C (байты), каждый элемент которого есть частное A[I] / B[I]. Деление двух чисел по алгоритму без восстановления остатка оформить как подпрограмму.

13. Задан массив ненормализованных чисел в формате с ПТ. Сформировать массив нормализованных чисел с ПТ. Нормализацию оформить в виде подпрограммы.

14. Заданы два массива положительных чисел A и B. Сформировать массив C, каждый элемент которого – наибольший общий делитель (НОД) двух чисел A[I] и B[I]. Вычисление НОД выполнить, используя алгоритм Евклида, и оформить как подпрограмму.

Указание. Числа A и B сравниваются между собой. Из большего числа вычитается меньшее. Сравнение и вычитания повторяются до выполнения условия A = B.