

Проектирование процессор ЭВМ

1 слайд

Добрый день! Я, студент группы Б17-503, Яковенко Иван представляю вам курсовой проект на тему "Проектирование процессора ЭВМ"

Курсовой проект «Проектирование процессора ЭВМ»

Студент группы Б17-503 Яковенко И. А.
Руководитель Ядыкин И. М.

2 слайд

Работа ведется с четырехразрядными числами в дополнительном коде со знаком
Спроектированный процессор выполняет 4 операции:

- УМНОЖЕНИЕ чисел со знаком с младших разрядов
- ПЕРЕСЫЛКУ ОТРИЦАТЕЛЬНУЮ – то есть взятие отрицательного значения модуля числа. Так же устанавливается признак результата равный знаку результата (1 если число меньше нуля, 0 если число равно нулю)

И два перехода:

- Условный переход, если признак результата равен 1
- Безусловный переход

Адрес перехода задается в виде смещения в дополнительном коде.

Техническое задание № 19-15

Оперативная память – **16x8**

Регистровая память – **8x4**

Операнды – дробные числа в дополнительном коде

Слово = **4 разряда**

Формат команд:

Первый операнд команды хранится в РП. Адресация прямая

Второй операнд хранится в ОП (РА2=0 - прямая адресация, РА2=1 - постиндексная косвенная вариант 2)

Результат операции **УМНОЖЕНИЕ** записывается по адресу **второго операнда**.

Результат **ПЕРЕСЫЛКИ ОТРИЦАТЕЛЬНОЙ** по адресу **первого операнда**

Операции:

УМНОЖЕНИЕ – алгоритм умножения чисел в дополнительном коде с младших разрядов множителя

ПЕРЕСЫЛКА ОТРИЦАТЕЛЬНАЯ – дополнительный код абсолютного значения **второго операнда**

пишется по **адресу первого операнда**. То есть, модуль второго операнда берем со знаком минус (исключение 0). Устанавливается признак результата: 0 – (результат = 0), 1 – (результат < 0)

ПЕРЕХОД, ЕСЛИ 1 – продвинутый адрес в счетчике команд замещается адресом перехода, если значение PR = 1. Используется относительная адресация (в команде – смещение со знаком)

БЕЗУСЛОВНЫЙ ПЕРЕХОД – продвинутый адрес в счетчике команд замещается адресом перехода.

Используется относительная адресация (в команде указывается смещение со знаком)

2

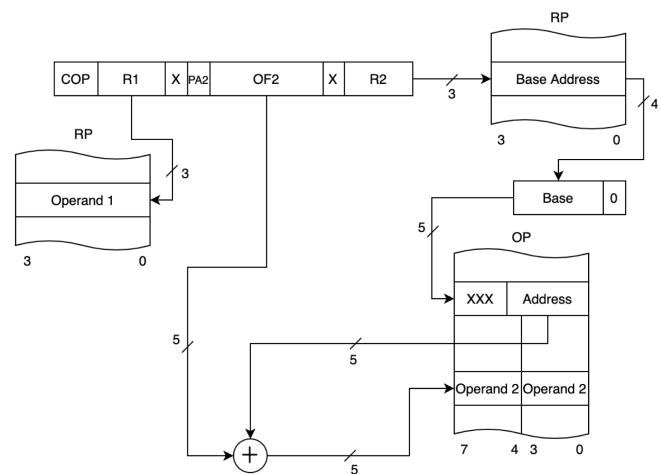
3 слайд

Для размещения команд в памяти были разработаны представленные форматы. Первый операнд команды УМНОЖЕНИЕ и ПЕРЕСЫЛКА ОТРИЦАТЕЛЬНАЯ задается при помощи прямой регистровой адресации. Второй операнд, в зависимости от признака адресации РА2, указывается либо с помощью прямой адресации, либо с помощью постиндексной косвенной. Для команд перехода используется относительная адресация и соответственно смещение со знаком, которое указывается в команде. Схема постиндексной косвенной адресации показана справа на слайде. Для формирования исполнительного адреса выполняется последовательное считывание из РП, ОП и сложение со смещением из команды.

Форматы команд и способы адресации

		RK1	RK2	RK3	RK4	
Переход		2 1	5			
		COP X	SM			
		15 14 13 12	8			
PA2 = 1 Умножение / Пересылка		2 3 1 1	5	1	3	
		COP R1 X PA2	OF2 X R2			
		15 14 13 11 10 9 8	4 3 2 0			
PA2 = 0 Умножение / Пересылка		2 3 1 1	5			
		COP R1 X PA2	A2			
		15 14 13 11 10 9 8	4			

Форматы команд



Постиндексная косвенная
адресация

3

4 слайд

Для адресации в БМК был выбран принудительный способ указания следующего адреса. Следующий адрес при отсутствии ветвления равен адресу указанному в команде. Если же необходимо ветвление переход осуществляется по А+1. Логическая схема проверки маски признаков представлена слева на слайде. В случае ветвления, адрес перехода всегда должен быть четным.

Блок выработки микрокоманд

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A3	A2	A1	A0	COP1	COP0	PA2	PR	YC15	YC14	YC13	YC12	YC11	YC10	YC9	YC8	YC7	YC6	YC5	YC4	YC3	YC2	YC1	SNO

Формат команды

Принудительная адресация

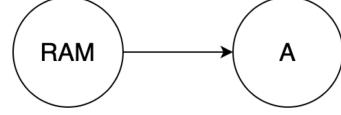
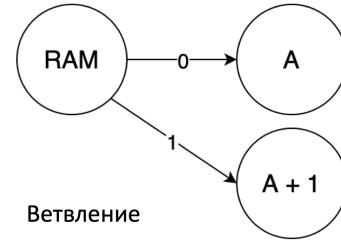
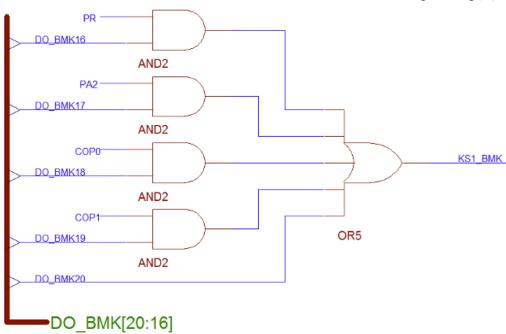


Схема формирования младшего разряда

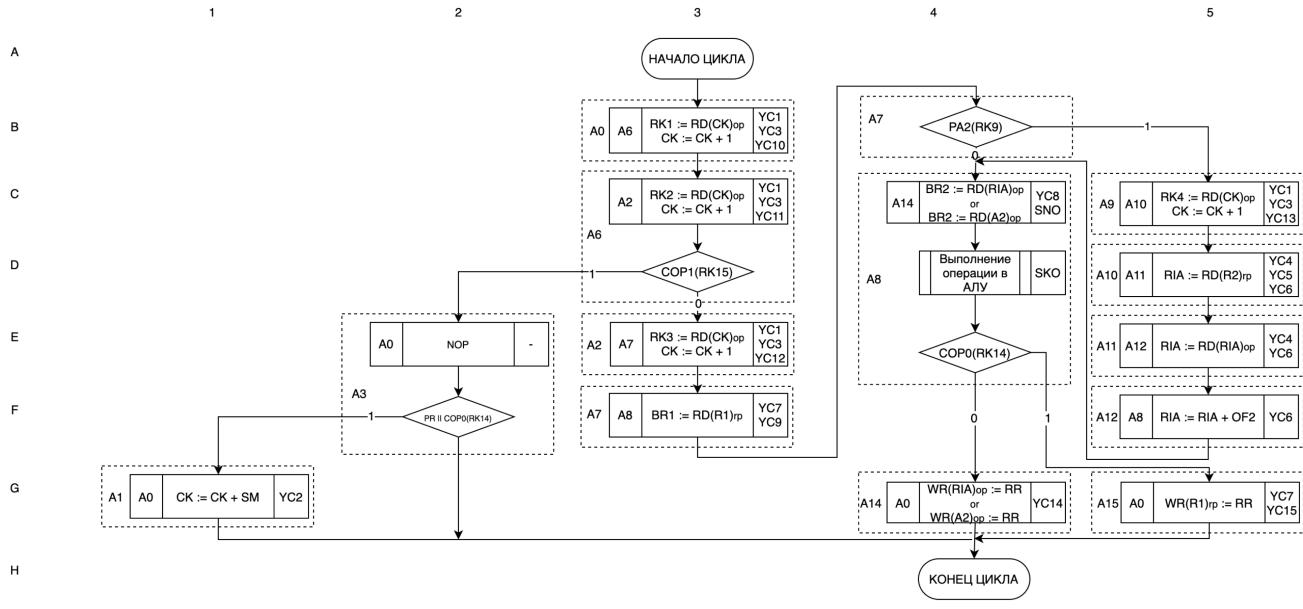
Последовательное выполнение

4

5 слайд

Далее перейдем непосредственно к алгоритму выполнения команд. Тут я бы хотел остановиться на способе реализации перехода по трем ветвям. Для этого была введена пустая команда (без управляющих сигналов). По умолчанию в алгоритме читаются два слова из оперативной памяти, далее идет определения типа команды – линейная / нелинейная. Далее в случае линейной команды читаются одно или два слова перед записью операндов в буферные регистры. Количество определяется способом адресации. Запись результата так же в зависимости от способа адресации производится в РП или ОП.

Алгоритм выполнения команд

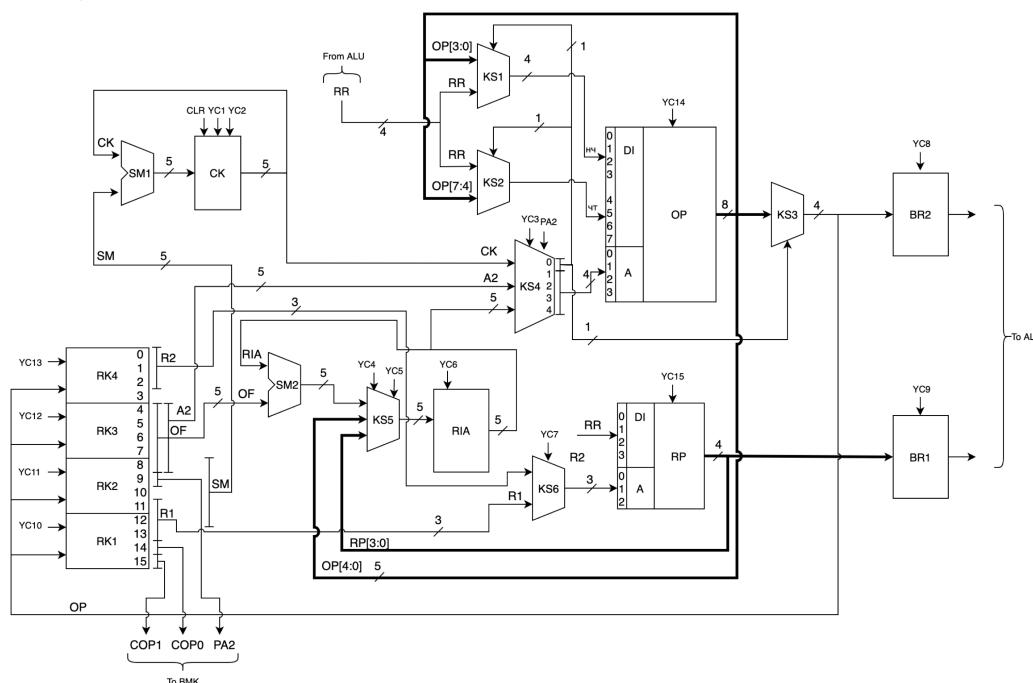


5

6 слайд

На схеме представлена функциональная схема блока управления командами. Здесь я хотел бы остановиться на формировании исполнительного адреса при использовании постиндексной косвенной адресации. Был введен регистр исполнительного адреса RIA. В него последовательно записывается адрес из регистровой памяти, расширенный нулем, базовый адрес из оперативной памяти и наконец сумма базы и смещения для обращения к оперативной памяти по исполнительному адресу. Два операнда, считанные из РП и ОП, поступают в буферные регистры BR1 и BR2, а далее в АЛУ.

Функциональная схема БУК

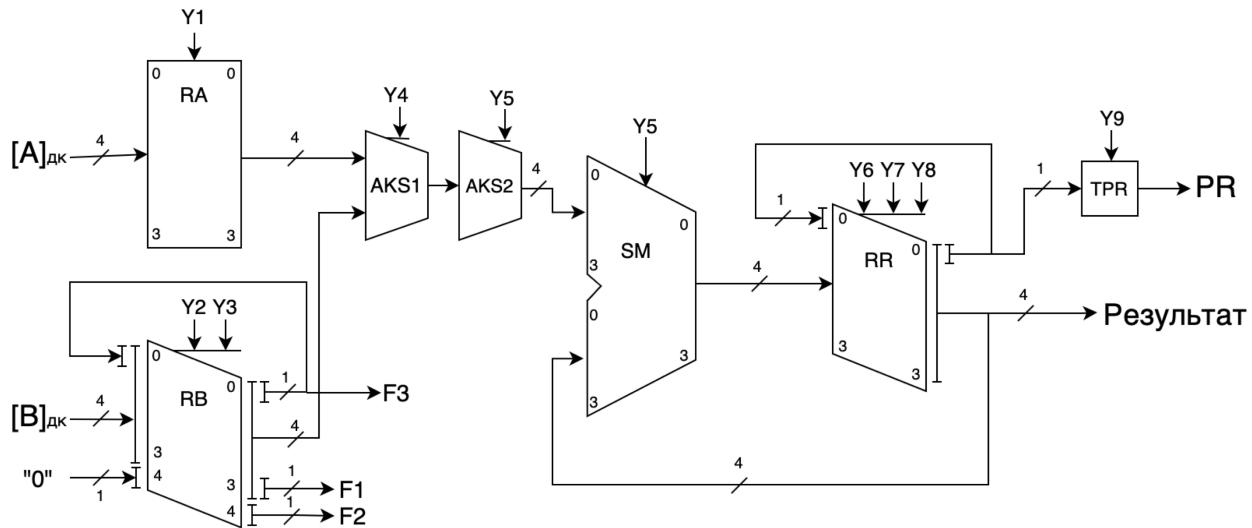


6

7 слайд

Переходя к АЛУ, необходимо рассказать про устройство, где сосредоточена основная логика работы АЛУ – блок операций. На функциональной схеме видно, что две операции (умножение и пересылка) были объединены с помощью КС1 которая выбирает operand на вход сумматора. КС2 предварительно инвертирует operand при заданном Y5. Так как работа идет в дополнительном коде, при инвертировании необходимо прибавлять 1. Это происходит автоматически с подачей Y5 на CI сумматора. Знак результата при второй операции сохраняется в TPR.

Функциональная схема БО

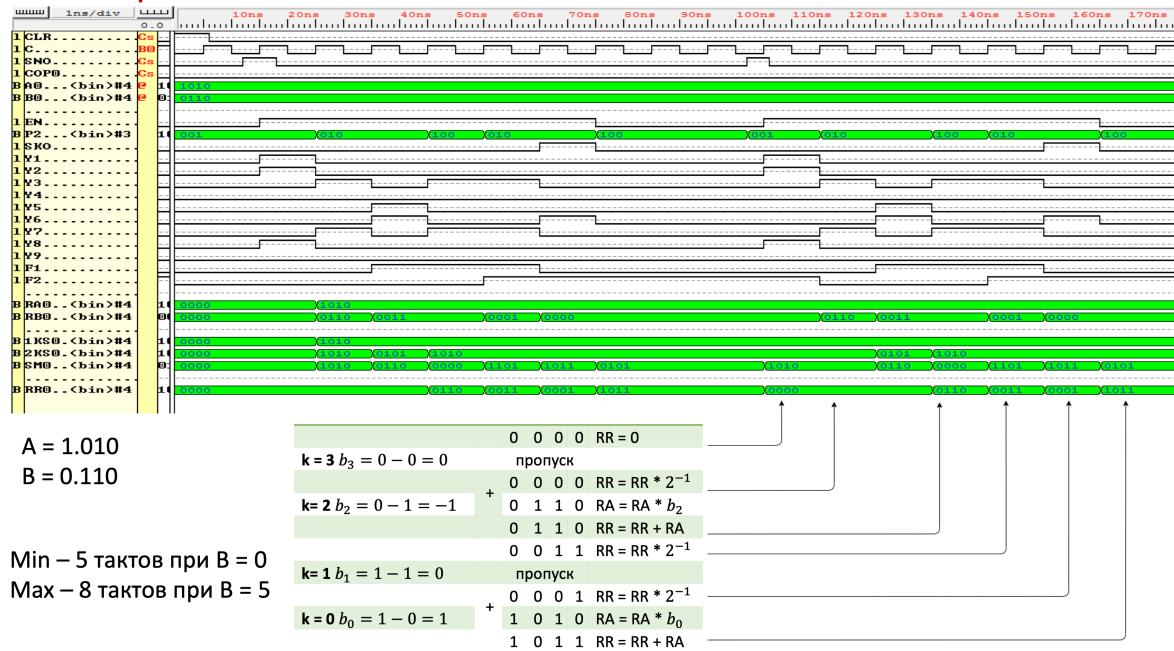


7

8 слайд

На слайде показан пример работы АЛУ на функциональном моделировании. Операнд A = 1.010 или -6/8, операнд B = 0.110 или 6/8. При перемножении ожидаемый результат должен быть -36/64 или при обрезке до 4 битов -5/8 то есть 1.011. Этапы вычисления проиллюстрированы с помощью примера вычисленного вручную. Для проверки старта МУУ не с нулевого состояния тести был запущен дважды. В процессе тестирования было выяснено, что минимальное число тактов достигается при операнде B = 0. Максимальное число тактов при B = 5.

Тестирование АЛУ



8

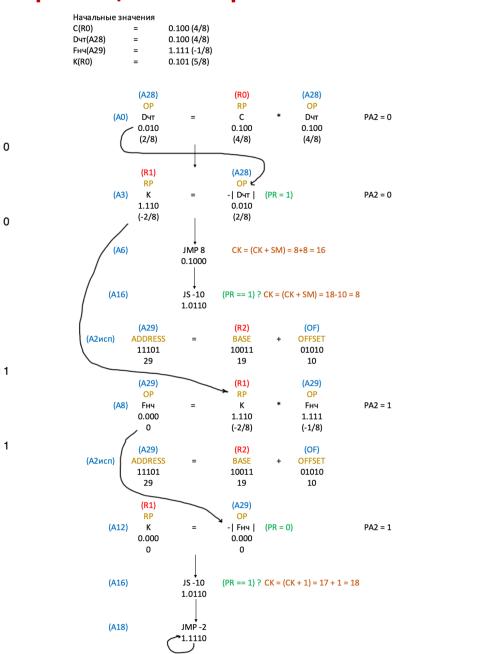
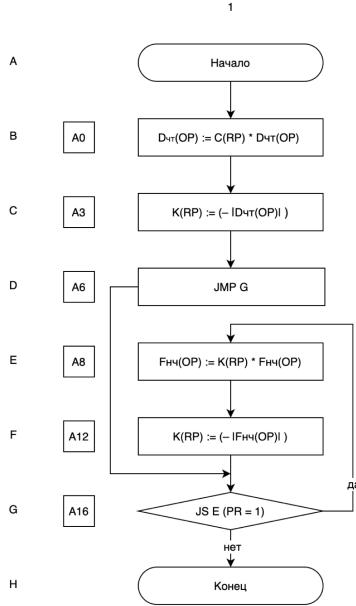
9 слайд

Для отладки процессора было проведено его тестирование. Данный тест обеспечивает:

- Проверку всех микрокоманд алгоритма (проход по всем веткам алгоритма)
- Проверку считывания команд разного формата, расположенных по четным и нечетным адресам в оперативной памяти (напр., первое умножение и пересылка)
- Проверку считывания операндов из оперативной памяти, расположенных по четным и нечетным адресам (D и F)
- Проверку записи результата в оперативную память по четным и нечетным адресам (D и F, в операциях пересылки)
- Проверку считывания operandов из регистровой памяти (C/K)
- Проверку записи operandов в регистровую память (K)
- Проверку всех используемых способов адресации (все)
- Проверку правильности вычисления признака перехода (есть или нет переход на A16)
- Проверку правильности работы АЛУ (значения совпадают с рассчитанными вручную)

Размещение команды в памяти представлено на рисунке справа. Значения, вырабатываемые процессором полностью совпадают со значениями, рассчитанными вручную. Соответственно на рассчитанном примере показан ход выполнения программы. Изогнутыми стрелками показана связь между записанными и считанными значениями. Изображен процесс вычисления исполнительного адреса для operandов с использованием постиндексной косвенной адресации.

Тестирование процессора



Пример выполнения тестовой программы

Адрес ячейки	Адрес слова	Команды и данные	Двоичный код	HEX
0	A0; A1	MUL, R0, 0, PA2, A28	0000 0001	0 1
1	A2; A3	A28, SND, R1	1100 0100	C 4
2	A4; A5	R1, 0, PA2, A28	1001 1100	9 C
3	A6; A7	JMP, 0, 8	1100 1000	C 8
4	A8; A9	MUL, R1, 0, PA2, OF	0000 1010	0 A
5	A10; A11	OF, 0, R2	1010 0010	A 2
6	A12; A13	SND, R1, 0, PA2, OF	0100 1010	4 A
7	A14; A15	OF, 0, R2	1010 0010	A 2
8	A16; A17	JS, 0, -10	1001 0110	9 6
9	A18; A19	JMP, 0, -2	1101 1110	D E
10	A20; A21	0000, 0000	0000	0 0
11	A22; A23	0000, 0000	0000	0 0
12	A24; A25	0000, 0000	0000	0 0
13	A26; A27	000, 19	0001 0011	1 3
14	A28; A29	[D] = +4/8, [F] = -1/8	0100 1111	4 F
15	A30; A31	0000, 0000	0000	0 0

Размещение программы в ОП

Адрес ячейки	Адрес слова	Команды и данные	Двоичный код	HEX
0	R0	[C] = +4/8	0100	4
1	R1	[K] = +5/8	0101	5
2	R2	A13	1101	D
3	R3	0000	0000	0
4	R4	0000	0000	0
5	R5	0000	0000	0
6	R6	0000	0000	0
7	R7	0000	0000	0

Размещение программы в РП

9

Заключение

Для каждого из спроектированных блоков было выполнено моделирование и тестирование. Блоки были соединены вместе, образуя процессор. В качестве дальнейшего развития проекта предлагается включение новых команд для процессора таких как СТОП, возможно написание своего рода транслятора из более понятного человеку кода в машинные слова.