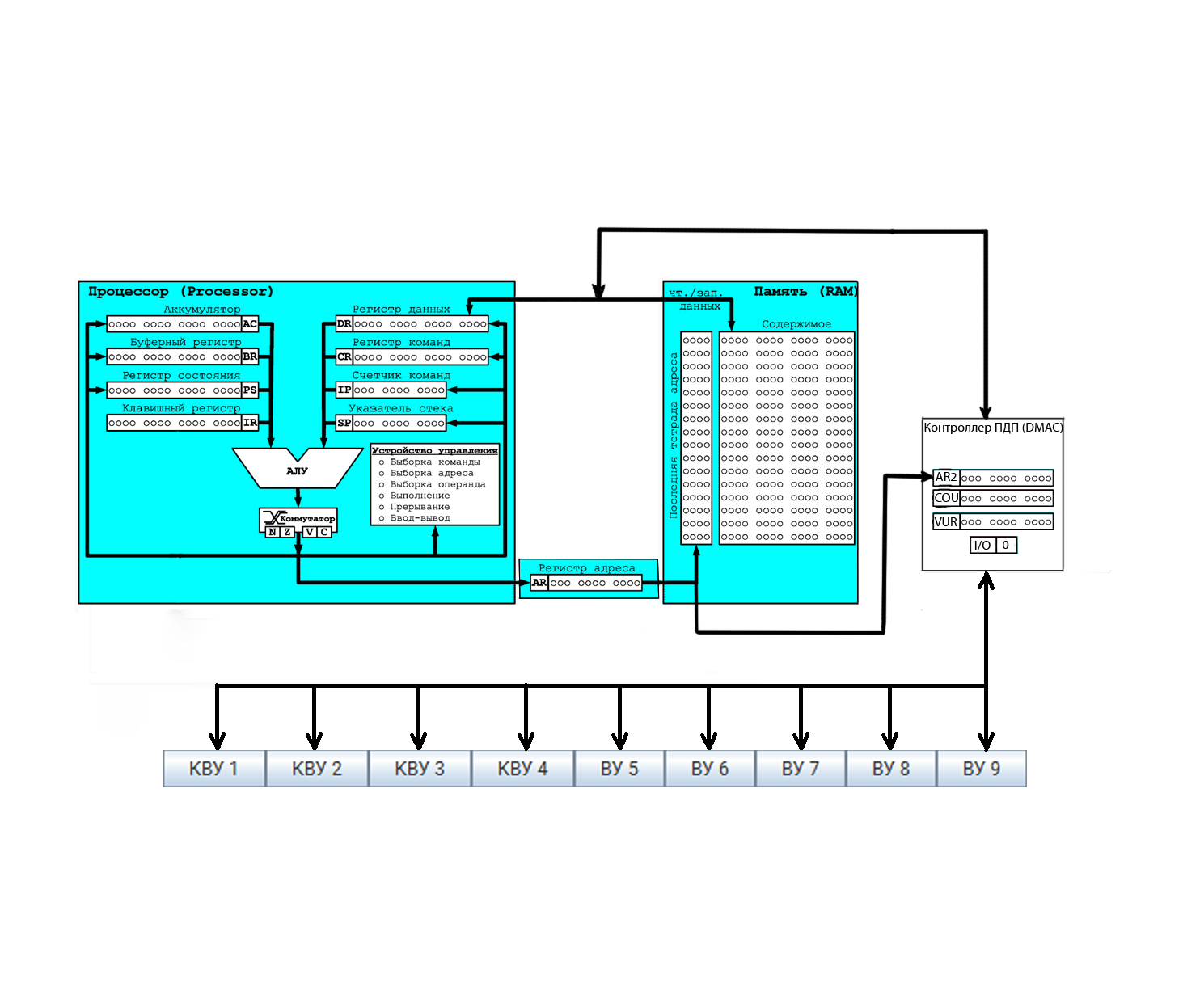
**Дополнительное задание**

**Задание:** Разработать контроллер прямого доступа для БЭВМ

Пример реализации:



1. Работает это чудо следующим образом:

у нас имеется контроллер прямого доступа к памяти, у которого есть 4 регистра.

1. Регистр адреса памяти (регистр с которого мы начинаем передачу данных)

2. Счетчик (кол-во подряд идущих ячеек, которые мы передаем на ву)

3. Номер Внешнего устройства, с которым мы работаем

4. Ввод/вывод (1/0)

На БЭВМ мы подаем команду и заполняем данные для работы контроллера ПДП, затем процессор спокойно занимается своими делами, пока контроллер не выполнит всю свою работу. После выполнения работы контроллером, он подает прерывание на процессор и это прерывание как-либо обрабатывается.

2. Можно было бы контроллер ПДП встроить в каждый контроллер внешнего устройства. Тогда, в принципе, можно было бы обращаться к памяти сразу несколькими внешними устройствами, и они могли бы работать параллельно. Но я посчитал это не совсем разумным, т.к. основная задача прямого доступа к памяти в том, что мы освобождаем от бесполезной нагрузки процессор и во время работы внешних устройств он может заниматься своими делами. НО для того, чтобы не было ситуаций, когда процессор и контроллер ПДП обращаются к памяти одновременно, у нас существует арбитр шины, который решает, чья очередь первая. Обычно арбитр отдает предпочтение сначала устройствам ввода/вывода, т.к. работу дисков и других движущихся устройств нельзя прерывать (это может привести к потерям данных). Получается, если работает какое-то устройство вводы-вывода, оно будет обращаться к памяти каждый раз и захватывать цикл памяти (работа процессора будет приостанавливаться на время, пока ву работает с памятью) Поэтому, если у нас будет много внешних устройств, которые независимо друг от друга будут обращаться к шине памяти, то процессор будет проводить много времени в ожидании (если ему нужно обращаться к памяти) и смысла от ПДП будет гораздо меньше.

Также из недостатков данной реализации – обращения к памяти внешних устройств также могут пересекаться, что усложняет логику работы арбитра шины.

Конечно, во внешних устройствах БЭВМ у нас ничего не крутится и не вертится, и мы можем иначе организовать арбитра шины (например, отдавать предпочтение сначала процессору) и встраивать ПДП в контроллеры ВУ.

Минусом реализованного аппаратно-управляемого ввода-вывода является то, что нам требуются значительно более сложные контроллеры (контроллер ПДП).

**Код программы:**

ORG 0x2

**V1:** WORD $INT1, 0x180

**A:** WORD 18

**B:** WORD 14

**COUNT:** WORD 0x0 ; целая часть деления

**RE:** WORD 0x0 ; остаток от деления

**ODZpl:** WORD 20

**ODZmi:** WORD 0xFFEC

**ISNEGRES:** WORD 0

**START:** EI

LD B

PUSH

CALL checkODZ ; проверка ОДЗ и исключений для числа В

POP

LD A

BEQ EXIT ; если А равно 0

PUSH

CALL checkODZ ; проверка ОДЗ и исключений для числа А

POP

LD A ; блок проверки знака результата операции

CALL ISNEG

LD B

CALL ISNEG

LD A ; abs(A)

CALL ABS

ST A

LD B ; abs(B)

CALL ABS

ST B

**WHILE:** LD A ; реализация деления A на B

SUB B

ST A

BEQ ENDLN1

BMI ENDLN2

LD COUNT

INC

ST COUNT

JUMP WHILE

**ENDLN2:** ADD B ; окончание в случае наличия остатка

ST RE

JUMP ENDLN

**ENDLN1:** LD COUNT ; окончание в случае, если остаток 0

INC

ST COUNT

ENDLN: LD ISNEGRES

BEQ EXIT

LD COUNT

NEG

ST COUNT

LD RE

NEG

ST RE

**EXIT**: CLA

OUT 0x2

HLT

**I1:** LD #0x1 ; исключительный случай 1

INT 1 ; (деление на 0)

HLT

**I2:** LD #0x2 ; исключительный случай 2

INT 1 ; (выход за границы ОДЗ)

HLT

**checkODZ:** LD &1

BEQ I1 ; подпрограмма, проверяющая ОДЗ и наличие

BPL PLUSB ; исключительных ситуаций

NEGB: CMP ODZmi

BMI I2

RET

PLUSB: CMP ODZpl

BEQ ALL

BPL I2

ALL: RET

**ISNEG:** BMI ISTR ; подпрограмма, выставляющая флаг,

RET ; определяющий знак частного

ISTR: LD ISNEGRES

BEQ FLAGTRUE

CLA

ST ISNEGRES

RET

FLAGTRUE: INC

ST ISNEGRES

RET

**ABS:** BMI ABS1 ; подпрограмма, возвращающая модуль

RET ; числа, записанного в аккумулятор

ABS1: NEG

RET

ORG 0x100

**INT1:** NOP ; обработка прерывания в случае

OUT 0x2 ; возникновения исключительной ситуации

IRET