УЧЕБНЫЙ МИКРОПРОЦЕССОР TCPU И ЕГО ПРИМЕНЕНИЯ

Долинский М.С., Коршунов И.В., ГГУ им. Ф. Скорины
dolinsky@gsu.by

Введение

В работах [1-5] подробно представлены инструменты разработки программного и аппаратного обеспечения встроенных цифровых систем, созданные в Гомельском государственном университете им.Ф.Скорины. Системы HLCCAD [2] и Winter [3] позволяет проектировать, симулировать и отлаживать программное обеспечение. Кроме того, обе они позволяют проводить совместную отладку программного и аппаратного обеспечения, в том числе и мультипроцессорных систем. Нами созданы также средства автоматизации разработки моделей микропроцессоров/микроконтроллеров (МП/МК), базирующиеся на генерации исходных ассемблерных текстов моделей МП/МК, за счет чего достигается увеличение производительности симуляции в 10 и более раз. Необходимо отметить, что несмотря на наличие специально разработанных декларативно-алгоритмических языков описания ядра (PCDL – Processor Core Decription Language) и периферии (PPDL – Processor Peripherals Decription Language), для создания такой модели разработчик должен иметь определенные навыки разработки и отладки ассемблерных программ. Одним из средств проверки эффективности данной разработки послужило создание модели учебного микропроцессора TCPU. Данная работа содержит его описание и лабораторные работы, основанные на использовании созданной модели микропроцессора TCPU.

Микропроцессор TCPU

Микропроцессор TCPU содержит четыре блока памяти, два регистра общего назначения, два регистра косвенной адресации и регистры указатели PC и SP. Ниже приведены таблицы ресурсов.

Стек процессора расположен в отдельной памяти, вершина стека указывается регистром SP. Когда в стеке что-то есть, регистр SP содержит адрес ячейки, куда поместили значение последним. Когда в стеке ничего нет, регистр SP содержит значение 0x00. Т.е. адрес 0x00 лежит за пределами стека. Таким образом, хотя память для стека имеет размер 256 слов, стек имеет размер 255 слов.

Название памяти	Размер слова памяти (бит)	Размер памяти (слов)	Описание
Code	8	256	Кодовая память
Data	8	256	Память данных
Stack	8	256	Стек
Hidden	8	2	Виртуальная память для регистров-указателей

Название регистра	Размер регистра (слов)	Память, в которой расположен	Адрес	Описание
A	1	Data	0x00	Регистр общего назначения
B	1	Data	0x01	Регистр общего назначения
R1	1	Data	0x02	Регистр косвенной адресации
R2	1	Data	0x03	Регистр косвенной адресации
PC	1	Hidden	0x00	Счетчик команд
SP	1	Hidden	0x01	Указатель вершины стека

Команды пересылки данных

Код инструкции	Мнемоника	Размер команды (в байтах)	Формат	Описание
0x01	MOV A,d	2	0x01 0xdd	Занести в регистр A второй байт кода операции
0x02	MOV R1,d	2	0x02 0xdd	Занести в регистр R1 второй байт кода операции
0x03	MOV R2,d	2	0x03 0xdd	Занести в регистр R2 второй байт кода операции
0x04	MOV R1,A	1	0x04	Занести в регистр R1 байт из регистра A
0x05	MOV R2,A	1	0x05	Занести в регистр R2 байт из регистра A
0x06	MOV A,R1	1	0x06	Занести в регистр A байт из регистра R1
0x07	MOV A,R2	1	0x07	Занести в регистр A байт из регистра R2
0x08	MOV A,#a	2	0x08 0xaa	Занести в регистр A байт из ячейки памяти данных с адресом, указанным во втором байте кода операции
0x09	MOV A,@R1	1	0x09	Занести в регистр A байт из ячейки памяти данных с адресом, указанным в регистре R1
0x0A	MOV A,@R2	1	0x0A	Занести в регистр A байт из ячейки памяти данных с адресом, указанным в регистре R2
0x0B	MOV @R1,A	1	0x0B	Занести в ячейку памяти данных с адресом, указанным в регистре R1, байт из регистра A
0x0C	MOV @R2,A	1	0x0C	Занести в ячейку памяти данных с адресом, указанным в регистре R2, байт из регистра A
0x0D	MOV B,d	2	0x0D 0xdd	Занести в регистр B второй байт кода операции
0x0E	MOV A,B	1	0x0E	Занести в регистр A байт из регистра B
0x0F	MOV B,A	1	0x0F	Занести в регистр B байт из регистра A

Команды работы со стеком

Код инструкции	Мнемоника	Размер команды (в байтах)	Формат	Описание
0x10	PUSH A	1	0x10	Положить байт из регистра A в стек
0x11	POP A	1	0x11	Взять байт из стека в регистр A
0x12	PUSH @R1	1	0x12	Положить байт из ячейки памяти данных с адресом, указанным в регистре R1, в стек
0x13	POP @R1	1	0x13	Взять байт из стека и положить в ячейку памяти данных с адресом, указанным в регистре R1
0x1F	STRST	1	0x1F	Сбросить указатель вершины стека

Команды передачи управления

Код инструкции	Мнемоника	Размер команды (в байтах)	Формат	Описание
0x20	JMP met	2	0x20 0xaa	Безусловный переход к инструкции по адресу 0xaa
0x21	JAZ met	2	0x21 0xaa	Переход к инструкции по адресу 0xaa, если в регистре A ноль

Арифметические операции

Код инструкции	Мнемоника	Размер команды (в байтах)	Формат	Описание
0x30	ADD A,d	2	0x30 0xdd	Прибавить к значению в регистре A второй байт кода операции и положить результат в регистр A
0x31	SUB A,d	2	0x31 0xdd	Вычесть из значения в регистре A второй байт кода операции и положить результат в регистр A
0x32	ADD @R1,d	2	0x32 0xdd	Прибавить к значению в ячейке памяти данных с адресом, указанным в регистре R1, второй байт кода операции и положить результат в эту ячейку
0x33	SUB @R1,d	2	0x33 0xdd	Вычесть из значения в ячейке памяти данных с адресом, указанным в регистре R1, второй байт кода операции и положить результат в эту ячейку
0x34	ADD R1,d	2	0x34 0xdd	Прибавить к значению в регистре R1 второй байт кода операции и положить результат в регистр R1
0x35	SUB R1,d	2	0x35 0xdd	Вычесть из значения в регистре R1 второй байт кода операции и положить результат в регистр R1
0x36	ADD R2,d	2	0x36 0xdd	Прибавить к значению в регистре R2 второй байт кода операции и положить результат в регистр R2
0x37	SUB R2,d	2	0x37 0xdd	Вычесть из значения в регистре R2 второй байт кода операции и положить результат в регистр R2
0x38	ADD@R1,A	1	0x38	Прибавить к значению в ячейке памяти данных с адресом, указанным в регистре R1, значение из регистра A и положить результат в эту ячейку
0x39	SUB @R1,A	1	0x39	Вычесть из значения в ячейке памяти данных с адресом, указанным в регистре R1, значение из регистра A и положить результат в эту ячейку
0x3A	DAA	1	0x3A	Команда daa корректирует результат сложения в регистре A двух упакованных двоично-десятичных (BCD) чисел (по одной цифре в каждом полубайте), чтобы получить пару правильных упакованных двоично-десятичных цифр. Команда используется вслед за операцией сложения упакованных двоично-десятичных чисел.
0x3B	DAS	1	0x3B	Команда das корректирует результат вычитания в регистре A двух упакованных двоично-десятичных (BCD) чисел (по одной цифре в каждом полубайте), чтобы получить пару правильных упакованных десятичных цифр. Команда используется вслед за операцией вычитания упакованных двоично-десятичных чисел.

Логические операции

Код инструкции	Мнемо-ника	Размер команды (в байтах)	Формат	Описание
0x40	AND A,B	1	0x40	Логическое И значений из регистров A и B. Результат помещается в регистр A.
0x41	OR A,B	1	0x41	Логическое ИЛИ значений из регистров A и B. Результат помещается в регистр A.
0x42	XOR A,B	1	0x42	Исключающее ИЛИ значений из регистров A и B. Результат помещается в регистр A.
0x43	NOT A	1	0x43	Логическое НЕ байта из регистра A.

Другие команды

Код инструкции	Мнемоника	Размер команды (в байтах)	Формат	Описание
0x00	NOP	1	0x00	Пустая команда (ничего не делать)

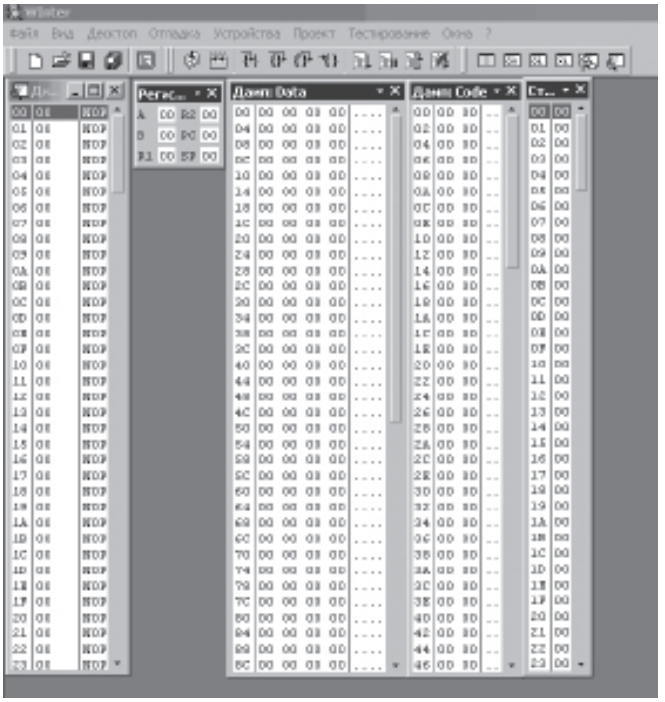
Лабораторные задания

Необходимо подчеркнуть, что корректность выполнения всех лабораторных заданий проверяется АВТОМАТИЧЕСКИ, с использованием сайта (DL.GSU.BY) дистанционного обучения ГГУ им Ф.Скорины и разработанных нами средств автоматизации проектирования программно-аппаратных систем [1-5].

Задание №1. Разработать программу (в машинных кодах) для решения поставленной задачи. Например:

Задание	Найти количество букв в слове (слово заканчивается символом *)
Формат ввода	R1 - адрес ячейки, содержащей первую букву слова
Формат вывода	A - количество букв в слове

Для проверки решения поставленной задачи студентам предоставляется среда Winter, настроенная на отладку программ в машинных кодах микропроцессора TCPU. Внешний вид среды отладки в данном случае выглядит следующим образом:

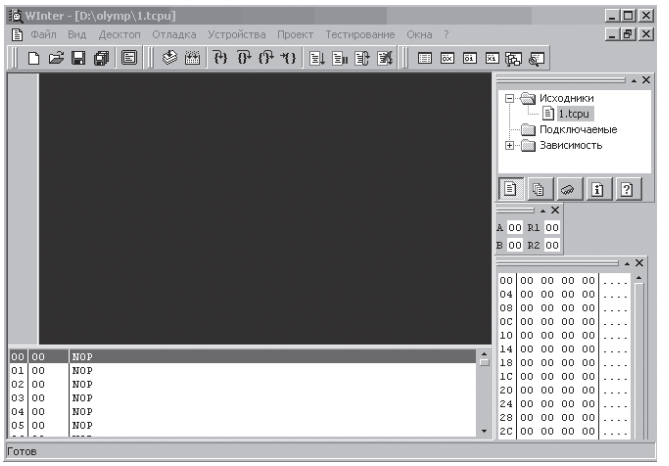


Студенты могут набирать непосредственно машинные коды команд, исполнять программу по командам, наблюдать изменение содержимого регистров, оперативной памяти и стека.

Задание №2. Разработать программу на ассемблере процессора TCPU для решения поставленной задачи. Например:

Задание	Найти произведение двух чисел
Формат ввода	R1 – первое число R2 – второе число
Формат вывода	A – результат

В этом случае среда WInter настроена на отладку программного обеспечения непосредственно на языке ассемблера процессора TCPU.



Задание №3. Задания на ассемблирование, например: дан текст программы на ассемблере TCPU,

```
        mov   A,R1
        push  A
        mov   A,R2
        push  A
met:
        mov   A,R1
        jaz   AR2
        mov   A,R2
        jaz   AR1
        sub   R1,1
        sub   R2,1
        jmp   met
AR1:
        pop A
AR2:
        pop A
```

требуется представить соответствующий машинный код для процессора TCPU в формате Intel Hex.

Задание №4. Задания на дизассемблирование, например:

дизассемблируйте следующий машинный код для процессора TCPU.

В задачах на дизассемблирование необходимо приведенный машинный код процессора TCPU перевести в ассемблерный код для этого процессора. Соответствующие мнемоники команд вписываются в поля ввода под машинным кодом. Мнемоники и имена регистров должны быть записаны большими буквами, между мнемоникой и первым операндом должен быть пробел. Если операндов два, после первого нужно ставить запятую, после запятой один пробел и второй операнд. Например:

MOV A, R2.

При записи ассемблерного кода команд перехода вместо имени метки нужно ставить адрес перехода как в машинном коде. Операнды всех команд, представляющие адреса нужно записывать в шестнадцатиричном формате в виде 0xdd (где x – латинское малое «икс»). Например:

JMP 0x1A.

При записи ассемблерного кода операнды команд, представляющие непосредственные значения, необходимо сначала переводить в десятичную форму и записывать без предваряющих нулей. Например:

0x01 0x05 – это MOV A, 5

0x36 0x30 – это ADD R2, 48.

Задание №5. Задание на проектирование схем – компонентов процессора TCPU (с использованием HLCCAD), например:

Имя HLCCAD проекта: TCPU.PRD

Входное устройство: Main

Название	Размерность	Тип
A	8	вход
MET	8	вход
PC_IN	8	вход
PC_OUT	8	выход

На входе даны: значение в регистре A (A), метка с адресом перехода (MET) и значение счетчика команд (PC_IN). На выходе должен быть адрес следующей команды.

Пример:

A : 00000000b
MET : 00000011b
PC_IN : 00000111b
PC_OUT : 00000011b

Задание №6. Задание на микропрограммную реализацию компонентов процессора TCPU (с использованием С-МПА, [4-5]), например:

Имя HLCCAD проекта: DAA.PRD

Входное устройство: DAA

Название	Размерность	Тип
A	8	вход
AO	8	выход
AF	1	выход
CF	1	выход

Десятичная коррекция двузначного упакованного BCD-числа. Старшие и младшие 4 бита входного числа A рассматриваются как цифры, т.е. значение 4-х бит шифрует цифру (например, 1001 = 9). Если младшие 4 бита превышают 9, то флаг AF устанавливается в 1, а эти биты корректируются. Если старшие 4 бита изначально или после корректировки младших превышают 9, то CF=1, и ставшие биты корректируются.

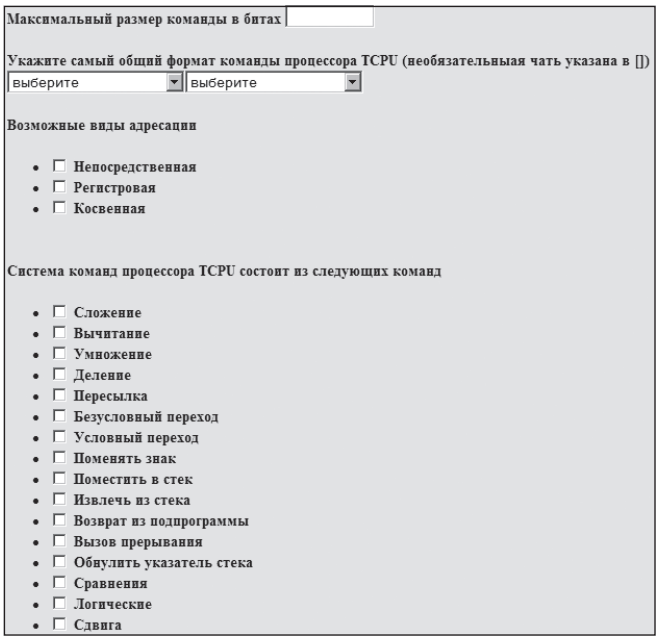
Примечание:

Например, если младшие 4 бита имеют значение 14, то в них надо оставить 4, и единицу прибавить к старшим 4-м битам, затем проверять и корректировать их.

Пример:

A: 11111111b
AO: 101
AF: 1
CF: 1

Задание №7. Тесты по теме, например:

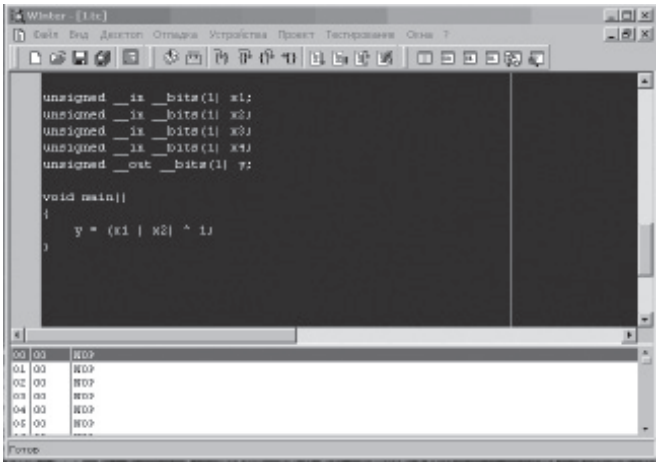


Задание №8. Разработка программ на С для процессора TCPU, например: найти произведение чисел A и B, результат занести в C.

Название	Размерность	Тип
A	8	вход
B	8	вход
C	16	выход

A – первый множитель, B – второй множитель, C – произведение.

Среда разработки при этом выглядит следующим образом:



Здесь система отладки Winter взаимодействует с разработанным нами С-компилятором для процессора TCPU.

В компиляторе С для TCPU поддерживаются следующие возможности:

Типы данных:

– int – целочисленный со знаком;

– unsigned – целочисленный без знака;

– одномерные массивы перечисленных типов.

При объявлении целочисленной переменной можно указать ее размерность. В силу 8-разрядности процессора TCPU размерность переменных не может превышать 8-ми бит. Если размерность не указана явно, то переменная будет иметь размерность 8 бит.

При вычислении выражений, содержащих переменные разных размерностей, используются следующие правила приведения типов:

– результат операции над двумя операндами имеет размерность большего операнда;

– результат операции со знаковым (int) и беззнаковым (unsigned) операндами является знаковым (int).

Пример описания переменных с указанием размерности:

– int __bits(8) in, __bits(3) out;

– unsigned __bits(1) bit_array[32].

Переменная in имеет размерность 8 бит, out – 3 бита. Массив bit_array состоит из 32 элементов, каждый имеет размерность 1 бит.

Константы.

Поддерживаются целочисленные и символьные константы. Символьные константы заключаются в одинарные кавычки, например, 'A' - символ 'A' или '\x10' – символ с шестнадцатеричным кодом 10. Целочисленные константы имеют размерность 8 бит. Можно указывать систему счисления целочисленных констант с помощью префиксов:

– восьмеричная система счисления, если число начинается с цифры '0', например, 0777 – число 777 в системе счисления с основанием 8;

– шестнадцатеричные начинаются с префикса '0x' или '0X', например, 0xff10;

– десятичные начинаются с любой цифры, кроме нуля.

Для беззнаковых констант используется суффикс 'U' или 'u', например, 40U.

Операторы.

Поддерживаются следующие операторы управления исполнением:

– for;
– if ... else;
– while;
– do ... while;
– break, continue;
– switch;
– вызов функций.

Арифметические операторы:

– =
– +, -, *
– &, |, ^, ~
– <<
– +=, -=, <<=, &=, |=, ^=
– ==, !=, <, >, &&, ||, !
– ++, --
– ? :
– оператор ',' (запятая)

Синтаксис и назначение перечисленных операторов полностью аналогичен языку Си.

Использование внутренних регистров и памяти TCPU.

Обращаться к внутренним регистрам и памяти TCPU можно через зарезервированные имена: _A, _B, _R1, _R2, _Data[0xff].

Например, найти количество букв в слове (слово заканчивается символом *).

Вход: R1 – адрес ячейки, содержащей первую букву слова.

Выход: A – количество букв в слове.

Решение:

```
void main()
{
        int i, j = _R1;
        for (i = 0;; i++)
        {
                int c = _Data[j++];
                if (c == '*') break;
        }
        _A = i;
}
```

При использовании внутренних регистров и памяти обязательно следует:

– в начале программы сохранить регистры во внутренние переменные, причем первым нужно сохранить R1;

– только в конце программы можно записать результат в соответствующий регистр.

Ограничения языка.

Обращение к массивам можно использовать только в операторах простого присваивания. Например:

```
int c = _Data[i];
c++;
_Data[i] = c;
```

Отсутствует поддержка:

– указателей;
– структур;
– операторов деления, сдвига вправо;
– возврат значений из функций.

Заключение

Описанные в данной работе лабораторные задания на базе модели микропроцессора TCPU активно используются в учебном процессе математического факультета Гомельского государственного университета им.Ф.Скорины и обеспечивают существенную интенсификацию и повышение качества обучения по соответствующим дисциплинам.

Литература:

1. Долинский М.С., Кугейко М.А. «Использование новых информационных технологий при обучении проектированию цифровых систем и программированию». – Минск, «Электроника-инфо», 2010, №4 (73). – с. 10-13.

2. Долинский М.С., Коршунов И.В. «Редактирование, симуляция и отладка аппаратного обеспечения с помощью HLCCAD». – Минск, «Электроника-инфо», 2010, №6 (75). – с. 22-26.

3. Долинский М.С., Коршунов И.В. «Среда WINTER для редактирования, симуляции и отладки программного обеспечения мультипроцессорных систем». – Минск, Электроника-ИНФО, 2011, №2. – с.53-56.

4. Долинский М.С., Коршунов И.В. «Технология разработки алгоритмически сложных цифровых систем с помощью автоматического синтеза микропрограммных автоматов». – Минск, Электроника-ИНФО, 2011, № 3. – с.53-57.

5. Долинский М.С., Коршунов И.В. «Автоматический синтез микропрограммных автоматов по С-МПА программам», 2012, №3. – с. 97-100.