Ассемблер для КР1801ВМ1 (БК0010 / БК0011 / УКНЦ / ДВК / PDP11)

January 16th, 16:53

Windows-версия и исходники: http://github.com/vinxru/pdp11asm

Представляю вам первую версию ассемблера для процессора КР1801ВМ1. Этот процессор совместим с системой команд PDP11 и используется в компьютерах БК001х и ДВК. Отдаю скомпилированным под Windows, с исходниками и его можно скомпилировать под другие ОС. В комплекте игра. Лицензия GPL, но как это оформить, я пока не понял. Эмулятор-отладчик можно скачать тут.

Когда то давно я уже писал этот ассемблер. В LJ можно найти про это заметку. Но та версия была потеряна и я не мог опубликовать её исходники, так как несколько файлов были использованы из закрытого проекта. Несколько дней назад ко мне обратился Lasoft со словами: "Видел что ты написал компилятор для Windows, респект! Можешь как-то помочь, может исходники? я хочу попробовать его и если что доработать". И я решил переписать ассемблер заново, да так, что бы он был кросс-платформенным. Наверное не стоило работать бесплатно, ради одного человека, который и не обещал пользоваться и вообще просил исходники. Ну да ладно. Всего два вечера работы. И быть может эта программа пойдет в резюме.

Я знаю, что для PDP-11 стандартом является ассемблер MACRO11. Я его не видел и не трогал. И уверен, что мой ассемблер сильно не совместим с ним. Многое пока не реализовано. Из арифметики поддерживается только сложение и вычитание. Без скобок. Нет макросов, нет условной компиляции, нет include.

Ассемблер формирует BIN файл, который умеет запускать эмулятор B2M. Первые два слова - адрес загрузки, длина. Далее идет двоичный код. А так же формирует LST файл, который эмулятор B2M использует для отладки. В отладчике будут отображаться имена переменных и комментарии, что очень сильно облегчает работу. Но! Почему то LST файл используется только для самых первых строк программы, дальше эмулятор всё забывает. Надо разбираться.

Если кому то потребуются новые возможности, я безвозмездно доработаю ассемблер. А пока пусть будет так. Ну действительно, зачем дорабатывать то, что никому не будет нужно.

Ниже следует краткое описание всех команд ассемблера:

Команды программы

ORG N .LINK N	Устанавливает адрес, куда будет выводится код после этой команды. По умолчанию используется адрес 0.
DB .BYTE	Записать байты. Через запятую можно указывать числа в формате: 10 - десятичный, 010 - восьмеричный, 0x10 - шестнадцатеричный, 10b - десятичный, 'A' - код символа, "ABCD" - несколько кодов символов. Так же поддерживается конструкция N DUP(C) записывающая N раз байт C.
DW .WORD	Записать слова.

.ASCII /text/	Вывести текст.
.END	Игнорируется
.BLKB N	Вывести N байт 0.
.BLKW N	Вывести N слов 0.
name EQU value name = value	Установить константу. Во всем файле name будет автоматически заменяться на value.
name:	Метка. Во всем файле name будет заменяться на адрес метки.
N:	Числовая метка. Область действия метки ограничивается символьными метками. Такая метка может быть использована только в командах условного перехода B? и SOB.
MAKE_BK0010_ROM "filename", start, end	Сформировать файл поддерживаемый эмулятором B2M. start, end - начальный и конечный адрес программы. Можно использовать метки.
MAKE_BINARY_FILE "filename", start, end	Выгрузить образ памяти в файл. start, end - начальный и конечный адрес. Можно использовать метки.
INSERT_FILE "filename", offset, size	Вставить двоичный файл. Можно вставить часть файла, задав размер и смещение в исходном файле.
ALIGN N	Выровнять адрес следующей команды на N-байт.
CONVERT1251TOKOI8R	Включить преобразование строк из кодировки 1251 в KOI8R.
CONVERT1251TOKOI8R OFF	Выключить преобразование строк из кодировки 1251 в KOI8R.
DECIMALNUMBERS	Далее все числа по умолчанию 10-ричные. В стандарте C.
DECIMALNUMBERS OFF	Далее все числа по умолчанию 8-ричные. В конце 10-чных чисел необходимо ставить точку.
INSERT_BITMAP1 "name", width, height	Вставить изображение преобразовав его в формат видеопамяти БК0010 2 цвета. Изображение должно быть в формате BMP 24 бита. width, height должны соответствовать размеру изображения в файле.
INSERT_BITMAP2 "name", width, height	Вставить изображение преобразовав его в формат видеопамяти БК0010 4 цвета. Изображение должно быть в формате BMP 24 бита. width, height должны соответствовать размеру изображения в файле.
INSERT_BITMAP1T "name", width, height	Вставить изображение преобразовав его в NAND и OR маски 2 цвета. В качестве прозрачного цвета используется #FF00FF.

INSERT_BITMAP2T "name", width, height	Вставить изображение преобразовав его в NAND и OR маски 4 цвета. В качестве прозрачного цвета используется #FF00FF.
; text // text	Комментарий
.	Адрес этой команды

Команды процессора без аргументов

HALT	Останавливает ЦП
WAIT	Останавливает ЦП до прерывания
BPT	Вызов прерывания
IOT	Вызов прерывания
EMT 0..255	Вызов прерывания
TRAP 0..255	Вызов прерывания
MARK 0..63	Вызов прерывания
RESET	Перезагрузка процессора
RTI	Выход из прерывания (Загружает PC, PS из стека)
RTT	Выход из прерывания пошаговой отладки.
NOP	Ничего не делает
CLC	Сбрасывает флаг C
CLV	Сбрасывает флаг V
CLZ	Сбрасывает флаг Z
CLN	Сбрасывает флаг N
CCC	Сбрасывает все флаги
SEC	Устанавливает флаг C
SEV	Устанавливает флаг V
SEZ	Устанавливает флаг Z

SEN	Устанавливает флаг N
SCC	Устанавливает все флаги
RET	Синоним RTS PC

Команды процессора с одним аргументом

(B) описании значит, что существует две версии команды CLRB - обрабатывающей байты и CLR - обрабатывающей слова.

JMP r	Переход по адресу
SWAB r	Поменять старший и младший байты местами
CLR(B) r	Записать 0
COM(B) r	Инверсия (заменть все биты на противоположные)
INC(B) r	Увеличить на единицу
DEC(B) r	Уменьшить на единицу
NEG(B) r	Изменить знак
ADC(B) r	Увеличить на единицу, если C=1
SBC(B) r	Уменьшить на единицу, если C=1
TST(B) r	Сравнить с нулем
ROR(B) r	Циклический сдвиг вправо через флаг C
ROL(B) r	Циклический сдвиг влево через флаг C
ASR(B) r	Сдвиг вправо, старший бит дублируется
ASL(B) r	Сдвиг вправо, младший бит равен 0
SXT r	Расширение знака. Если N=0 записывает 0, иначе -1
MTPS r	Установка регистра флагов

MFPS r	Чтение регистра флагов
CALL r	Синоним JSR PC,

Аргументы команды

R0, R1, R2, R3, R4, R5, SP, PC	Регистр. Все регистры 16 битные. 8 битные команды будут работать с младшими их половинами.
(REG) или @REG	Значение по адресу
@(REG)	Значение по адресу по адресу.
-(REG)	Значение по адресу. Уменьшить регистр до выполнения команды на 1 для байта. И на 2 для слова.
@-(REG)	Значение по адресу по адресу. Уменьшить регистр до выполнения команды.
(REG)+	Значение по адресу. Увеличить регистр после выполнения команды на 1 для байта. И на 2 для слова.
@(REG)+	Значение по адресу по адресу. Увеличить регистр после выполнения команды.
IMM(REG)	Значение по адресу, который рассчитывается как сумма регистра и числа.
@IMM(REG)	Значение по адресу по адресу, который рассчитывается как сумма регистра и числа.
#A	Число или адрес переменной.
@#A	Значение по адресу или переменная. Абсолютная адресация.
A	Значение по адресу или переменная. Адрес относительно PC.
@A	Значение по адресу по адресу. Первый адрес относительно PC.

Команды процессора с двумя аргументами

MOV(B) a, b	Скопировать
CMP(B) a, b	Сравнить A и B
BIS(B) a, b	Логическое ИЛИ

BIC(B) a, b	Логическое И-НЕ (B = B & ~A)
BIT(B) a, b	Логическое И без сохранения результата
ADD a, b	Сложение
SUB a, b	Вычитание

Команды условного перехода

Могут переходить только в пределах -256..+254 байта, относительно адреса следующей команды. Числа указанные в аргументе команды интерпретируются как числовые метки.

BR imm	Всегда
BNE imm	Не равно (Z=0)
BEQ imm	Равно (Z=1)
BGE imm	Больше равно для знаковых типов (N^V=0)
BLT imm	Меньше для знаковых типов (N^V=1)
BGT imm	Больше для знаковых типов (Z|(N^V)=0)
BLE imm	Меньше равно для знаковых типов (Z|(N^V)=1)
BPL imm	Результат положительный (N=0)
BMI imm	Результат отрицательный (N=1)
BHI imm	Больше (C|Z=0)
BVC imm	Нет знакового переполнения (V=0)
BVS imm	Знаковое переполнение (V=1)
BHIS imm / BCC imm	Больше или равно / нет переполнения (C=0)
BLO imm / BCS imm	Меньше / переполнение (C=1)

Необычные команды процессора

JSR reg, a	Вызвать подпрограмму (REG - только регистр, A - любой аргумент)
RTS reg	Выход из подпрограммы (REG - только регистр)
XOR reg, a	Исключающее ИЛИ (REG - только регистр, A - любой аргумент)
SOB reg, imm	Цикл (REG - только регистр, IMM - адрес). Числа указанные в аргументе IMM команды интерпретируются как числовые метки.

Tags: программирование, ретро