ZMAC v1.6 был экспериментальной версией, которая никогда не выпускалась.

ZMAC v1.7 содержит множество улучшений и несколько незаметных исправлений. Улучшения главным образом направлены на то, чтобы заставить ZMAC принять исходный код, написанный для других ассемблеров. Особый интерес представляет исходный код в общественном достоянии, который вы хотите ассемблировать с минимальным редактированием. Альтернативные операторы выражений, свободная форма исходных строк, расширенный синтаксис макросов и .PRINTX, и снятие ограничений на использование имен регистров для символов - попадают в эту категорию. Возможность объявлять глобальные однобайтовые величины требует взаимодействия между ассемблером и компоновщиком. ZMAC теперь обрабатывает внешние байты так же, как это делают ассемблеры SLR, и получающиеся модули REL могут быть соединены с помощью ZML v2.0. В следующем списке приводится краткое описание изменений, внесенных в ZMAC начиная с версии v1.5.

1. Новые операторы выражений
2. 'Новые' инструкции Z80/Z180
3. Свободная форма исходных строк
4. Теперь нет зарезервированных символов
5. Списки аргументов макросов
6. '^' в макро-вызовах
7. Расширенный синтаксис .PRINTX
8. Внешние байтовые величины
9. Использование регистров сообщений ZCPR
10. Сообщения об ошибках
11. Исправлена ошибка - Undefined Values (неопределенные значения)
12. Сохранение DS в ASEG
13. **Новые операторы выражений**

Ряд операторов, используемых в выражениях были добавлены в качестве альтернативы существующим. Это позволяет ZMAC принять синтаксис, разрешенный другими ассемблерами. Обратите внимание, что параметрами этих операторов являются символы или константы, не регистры. В примере VAL и MASK символы, определенные в программе как локальные или внешние величины. Новые и существующие операторы абсолютно эквивалентны.

|  |  |  |  |
| --- | --- | --- | --- |
| **Новый** | **Существующий** | **Функция оператора** | **Пример** |
| >> | SHR | Сдвиг вправо N раз | VAL SHR 8 |
| << | SHL | Сдвиг влево N раз | 1 SHL 4 |
| # | XOR | Побитовое исключающее ИЛИ | VAL # MASK |
| | | OR, ! | Побитовое ИЛИ | VAL | MASK |
| ~ | NOT | Побитовая инверсия | VAL AND ~ MASK |
| & | AND | Побитовое И | VAL & MASK |

Этот список новых операторов добавляет и расширяет список в Главе 4 (Выражения) Руководства ZMAC. Обратите внимание, что использование '!' в качестве альтернативы для OR было всегда возможно, но не было включено в руководство.

1. **Малоизвестные инструкции Zilog**

ZMAC теперь ассемблирует малоизвестную инструкцию "IN F,(C)", которая производит объектный код ED 70. Эта инструкция заставляет данные из порта с номером в регистре C быть считанными в байт флагов ('F' в 'AF'). Хотя эта инструкция редко встречается, она может использоваться в низкоуровневых подпрограммах интерфейса устройств, как модем или контроллер памяти на основе порта. Zilog и Hitachi упоминают о существовании этой инструкции, но почти ничего не говорят о ней. ZMAC отмечает любую другую попытку использовать регистр F как ошибку, потому что нет соответствующих определенных инструкций процессора. Обратная операция, которая могла бы быть "OUT (C),F", не предусмотрена процессором. Большинство других ассемблеров не ассемблируют эту инструкцию, сообщая о ней как об ошибке.

1. **Свободная форма исходных строк**

Исходные строки языка ассемблер используют свободный формат, теперь нет никаких установленных ограничений на позиции в которых начинаются поля метки, оператора и операнда. Все следующие примеры будут ассемблироваться правильно.

v---- первая колонка

DOSYM: LD (SYMVAL),HL ; Значение символа

LD (SYMTYP),A ; и его атрибуты

DOSYM:LD (SYMVAL),HL ; Значение символа

LD (SYMTYP),A ; и его атрибуты

DOSYM: LD (SYMVAL),HL ; Значение символа

LD (SYMTYP),A ; и его атрибуты

; ':' является необязательным, пока не возникает неоднозначность.

DOSYM LD (SYMVAL),HL ; Значение символа

Обратите внимание, что возникнет неоднозначность, если первая лексема является кодом операции которая предназначена для использования в качестве метки:

LDIR ADC A,B ; Ошибка. Поле оператора неоднозначно

LDIR: ADC A,B ; LDIR интерпретируется как метка

1. **Теперь нет зарезервированных символов**

Более ранние версии резервировали имена регистров (A, B, C, D, H, L, AF, BC, DE и HL и т.д.) для использования только там, где ожидается регистр. Вы не могли использовать их в качестве меток или определить их с помощью операторов EQU (или аналогичных). Версия 1.7 снимает это ограничение, и использование имен регистров больше не помечается как ошибка. Когда имя регистра используется в качестве метки или определенной величины, любая двусмысленность решается в зависимости от контекста. Например, следующий фрагмент кода иллюстрирует, как неоднозначность разрешена и как значение, присвоенное B, фактически используется (вместо регистра) в выражении.

B EQU 13 ; Присваивает значение символу B

LD B,41 ; Регистр B теперь содержит 41

LD A,B ; здесь, используется регистр

; и A будет содержать 41

LD A,B+1 ; значение 13 используется для B

; и A будет содержать 14

Почему вы хотите запутать ваш исходный код с именами регистров, используемых в качестве переменных? Ведь это может сделать источник сложным для восприятия. Это обычно в математических алгоритмах, используют одну букву в качестве переменной, и создают соблазн использовать одни и те же буквы при реализации алгоритма на ассемблере. Ранние программы SORT, используемые в DIR и программах FILEFIND, были написаны таким путем из алгоритмов, выраженных на языках как C, Pascal или RATFOR.

1. **Списки аргументов макросов**

Списки аргументов макросов могут теперь использовать пробелы, символы табуляции и запятые для разграничения (разделения) аргументов. Разделителем теперь считается строка, которая может содержать произвольное количество пробелов и символов табуляции и не более одной запятой. Такая строка эквивалентна одной запятой. Все списки аргументов в примерах ниже точно эквивалентны в отношении использования разделителя.

Примеры допустимых макроопределений (только первая строка)

SAMPLE MACRO ARG1,ARG2,ARG3,ARG4

SAMPLE MACRO ARG1, ARG2 ARG3 ARG4

SAMPLE MACRO ARG1 ARG2 , ARG3 ,ARG4

Примеры допустимых макро-вызовов

SAMPLE EXP1,EXP2,EXP3,EXP4

SAMPLE EXP1 EXP2 ,EXP3 EXP4

SAMPLE EXP1 , EXP2 EXP3, EXP4

Символы табуляции и/или пробелы иногда используются в макро-вызовах, чтобы выровнять параметры на странице для удобочитаемости. Посмотрите исходный код для ZCPR33 и источники в Z34RCP11.LBR в качестве примеров, в которых таблицы команд и другие структуры данных закодированы таким образом. В этих случаях, многие параметры в макро-вызовах являются опциями, которые выбрал установщик. Предыдущие версии ZMAC не могли ассемблировать макросы в Z34RCP11, не заменяя разделители пробелы запятыми. Эта версия ZMAC интерпретирует такие разделенные пробелами макросы без проблем.

1. **Символ ESC для макро-вызовов**

Для некоторых макро вызовов, обсуждение которых был пропущено в руководства ZMAC важным является символ ESC. Это символ стрелка вверх (^). При использовании в качестве реального аргумента в вызове макроса, он вызывает следующий символ для использования без какого-либо специального смысла. '^' сам добавляет себя, как в '^^'. Символами, которые имеют подобное особое значение в фактических параметрах макросов (аргументах), являются апостроф ('), процент (%), левые и правые скобки (< и >), и амперсанд (&). Следующий пример показывает один случай, где требуется использование ^.

Мы определяем тривиально простой макрос:

STRU MACRO ARG

DB ARG

ENDM

В программе мы хотим определить некоторые данные, используя этот макрос:

STRU 40H

который генерирует следующий исходный код:

DB 40H

Альтернативный способ определить те же данные был бы (без макроса)

DB 1<<6 ; То же самое, что 1 SHL 6

Используя макрос, чтобы сделать 'очевидное'

STRU 1<<6

производит

DB 1<6 ; один '<' был отброшен!

и это не входило в наши планы! Это потому, что макропроцессор отбрасывает первый '<', предполагая, что это является началом группы обрабатываемой как единое выражение.

Правильный способ вызова макроса следующий:

STRU 1^<^<6

который производит

DB 1<<6 ; То же самое, что 40H

Еще один правильный и точно эквивалентный вызов:

STRU <1 SHL 6>

который производит

DB 1 SHL 6 ; То же самое, что 1<<6

1. **Расширенный синтаксис .PRINTX**

У директивы ассемблера .PRINT (синонимом является .PRINTX) имеется новый расширенный синтаксис, надмножество форм, допускаемых ассемблерами M80, SLR и ZAS.

.PRINT [n,][QSTR[,QSTR...]] [;comment]

QSTR - заключенная в кавычки строка, как определено любым из вышеупомянутых ассемблеров, и n - является необязательным номером прохода, в котором строка(и) будут отображаться. Последняя (или только) строка в списке аргументов не должна быть заключена в кавычки, пока она не содержит ';'. См. PRINTX.Z80 (в Z80TEST.LBR) для примеров.

Размер буфера строки, которую отображает PRINT, был увеличен с 65 до 78 символов. Это наиболее близко к полной строке на экране терминала, отображаемой без автоматического переноса на новую строку, которую обеспечивают многие терминалы.

1. **Внешние байтовые величины**

Оригинальный формат Microsoft REL не предусматривал правил для величин с размером байта, определенных внешними. Позже, определения формата REL были расширены (без общедоступной документации в типичном стиле MicroSoft), для включения расширенных элементов REL. Одно из этих расширений используется для обработки байтовых величин. ZMAC v1.7 теперь распознает внешние байтовые величины и создает правильные элементы в файле REL для связывания редактором связей. Если вы решите использовать внешние величины с размером байта (или если источник, который вы ассемблируете делает это), то ZMAC v1.7 обработает их. Однако полученные mREL файлы должны быть связаны с помощью ZML v2 или выше или редакторами связей L80 Microsoft, LINK-80 DRI или SLR. Внешние байты не распознаются M80. Очевидно только языковые компиляторы Microsoft (Fortran, и т.д.) производят их для соединения с помощью L80. Файл BYTEXT.DOC содержит более полную информацию и примеры использования внешних байтов.

1. **Использование регистров сообщений ZCPR**

ZMAC теперь инициализирует только те регистры сообщений ZCPR3, присвоенные ему во время конфигурации с помощью ZCNFG. Это позволяет остальным регистрам сообщений быть использованными операционной системой или прикладными программами без помех от ZMAC. Предыдущие версии инициализировали все присвоенные пользователями регистры сообщений. Одним из вариантов использование для программы является сохранение текущего времени часов в регистрах сообщений, чтобы затем использовать сохраненное значение после выполнения другой программы для вычисления затраченного времени. Для корректной работы регистры, используемые для часов, не должны быть нарушены выполняющейся программой (ZMAC в этом случае).

1. **Сообщения об ошибках**

Исправление ошибки

Определение типа "VAR EQU 0D457", в котором отсутствует окончание "H", представляется как ошибка. Предыдущие версии не всегда обнаруживали эту ошибку.

Изменения в сообщениях об ошибках

Теперь выдается некритичная ошибка 'B' для выражений в макро-параметрах, которые используют не сбалансированные "<" и ">".

Ошибка 'K' больше не существует. Это была жалоба об использовании имен регистров в области метки и уже не актуальна (См "Теперь нет зарезервированных символов").

1. **Исправлена ошибка - Undefined Values (неопределенные значения)**

Неопределенные значения, помеченные как ошибка, показаны в листинге ассемблера со значением 0000. Предыдущие версии иногда показывали случайные значения в листинге PRN, противоречащие значению 0000 отправляемому в объект файл.

1. **Сохранение ASEG в mREL объектных файлах**

Размер ASEG ранее не включал указанное без инициализации место в памяти (пространство DS). У ассемблеров SLR есть тот же недостаток, но M80 объявляет такое пространство, как зарезервированное. ZMAC17 считает ASEG DS пространством, как будто оно было инициализировано, вынудив компоновщик уважать пределы ASEG, когда он выделяет начальный адрес для CSEG (или других сегментов). Это - неясный момент, и я не могу придумать ни какого использования для него. Но это логично ввиду абсолютной природы ASEG адресов. ZMAC17 соглашается с M80.