|  |  |  |  |
| --- | --- | --- | --- |
| Команда | Регистр для результата | Операнд 1 | Операнд 2 |
| ADD  (сложение) | rd | rs | rt |
| ADDI  (сложение с константой) | rd | rs | imm |
| OR  (побитовое ИЛИ) | rd | rs | rt |
| AND  (побитовое И) | rd | rs | rt |
| XOR  (исключающее ИЛИ) | rd | rs | rt |
| NOR  (ИЛИ-НЕ) | rd | rs | rt |
| SLL  (логический сдвиг влево) | rd | rs | rt |
| ROT  (циклический сдвиг влево) | rd | rs | rt |
| BNE  (условное ветвление, если НЕ равны) | rd | rs | rt |
| LD  (загрузка из памяти в регистр MR) | addr | | |
| ST  (сохранение из регистра MR в память) | addr | | |
| JMP  (безусловный переход по метке) | метка | | |
| JAL  (вызов подпрограммы по метке) | метка | | |
| JR  (безусловный переход по значению регистра) | rs | | |
| JALR  (вызов подпрограммы по значению регистра) | rs | | |
| NOP  (нет операции) | (количество повторений) | | |

**§1.Инструкции**

**1. Команда**

При записи команд можно использовать как строчные, так и прописные буквы латинского алфавита. После команды обязателен пробел.

**2. Регистры**

Для записи регистров в командах ADD – BNE, JR, JALR возможно использование имени или номера регистра, записав перед ним символ $.

|  |  |
| --- | --- |
| 0 | A |
| 1 | B |
| 2 | C |
| 3 | D |
| 4 | E |
| 5 | F |
| 6 | G |
| 7 | H |
| 8 | W |
| 9 | K |
| 10 | MR |
| 11 | LR |
| 12 | ZERO |
| 13 | T0 |
| 14 | T1 |
| 15 | T2 |

После записи первого и второго регистров следует ставить запятую, а затем пробел.

Пример:

and $C, $A, $b

Or $d, $C, $0

XOR $1, $c, $D

Запись константы при использовании команды ADDI возможна в двоичной, шестнадцатеричной и десятичной СС( при записи в двоичной перед числом следует записывать 0b, при записи в шестнадцатеричной – 0x, а при записи в десятичной приставка не требуется)

Пример:

# Запись числа 12 в шестнадцатеричной, двоичной и десятичной СС

addi $1, $E, 0xC

ADDI $0, $D, 0b1100

Addi $1, $2, 12

**3. Адрес**

При использовании команд LD, ST после пробела следует записывать адрес, представленный, так же, как и константы, в десятичной, двоичной или шестнадцатеричной СС с соответствующими приставками. Указывается младший адрес из двух.

Пример:

# Загрузка/выгрузка данных в 252 ячейку, а затем в 252+2=254 ячейку

LD 0xFC

St 0b11111100

Ld 252

**4. Метки**

При записи метки в программе, после неё ставят двоеточие. Используя команды JMP, JAL после пробела записывают только имя метки без каких-либо знаков.

Пример:

Nor $E, $F, $4

Jmp tut

Add $0, $1, $D

tut:

add $1, $A, $b

Xor $a, $0, $b

**5. Команда NOP**

При использовании команды «Нет операции», если она используется один раз следует написать только NOP, если требуется несколько повторений данной команды, то после пробела в скобках следует записать количество повторений.

Пример:

NOP

ADD $1, $B, $4

NOP (4)

**§2.Директивы**

Для облегчения написания программного кода введены некоторые директивы, программный код можно писать как с использованием директив, так и без, как было показано ранее. Для входа в директиву достаточно написать её название, начинающееся с символа точки.

**1. Директива .def**

Директива .def позволяет присвоить любому регистру новое имя, причём старое имя остаётся действительным, иными словами .def позволяет ссылаться на регистр через некоторое символическое имя. Назначенное имя может использоваться во всей нижеследующей части программы для обращений к данному регистру (для функций ADD - BNE, JR, JALR) . После входа в директиву на новой строчке указывается новое имя регистра, а через пробел - старое имя/номер регистра с символом $ вначале.

Пример:

Add Sum2, Sum1, $zero # тоже самое, что и add $A, $2, $zero

.def

Sum1 $2

Sum2 $A

**2. Директива .set**

Директива .set присваивает имени некоторое значение. Данная директива в нашем ассемблере используется лишь для функции ADDI. Ограничение – значение не должно превышать число 15. После входа в директиву на новой строчке указывается имя переменной, а через пробел – значение.

Пример:

Addi $a, $b, five

.set

five 0b101 # можно указывать чисто в 2-й, 16-й(0x5) и 10-й(5) СС

**3. Директива .data**

При помощи данной директивы можно записать данные в ячейку и присвоить адресу этой ячейки имя, которое впоследствии можно использовать в функциях LD и ST. Данные 32-х разрядные, могут быть представлены в 2-й, 16-й и 10-й СС. После входа в директиву на новой строчке указывается новое имя адреса, а через пробел – данные.

Пример:

Ld 1sum # В регистр MR загрузится значение 125

St 2sum # В 2sum запишутся данные из MR, вместо 334 там будет 125

.data

1sum 125 # или в 16-й 0x7D, или в 2-й 0b1111101

2sum 334 # или в 16-й 0x14E, или в 2-й 0b101001110

**4. Директива .text**

В ней описывается только код по всем правилам описанным в §1.Инструкции, директива .text нужна для того, что бы была возможность переходить из одной директивы в другую.

Например:

.text

add $a, $3, $zero # Складываем значение регистров $3 и $zero

addi $1, $2, p1 # Прибавляем к регистру $2 значение p1 = 8 (.set)

ld res # Загружаем в MR значение 455 (res из .data)

.data

res 455

.text

addi $MR, $MR, sum # MR + 15 (sum = 15 из .set) теперь в MR число 470

st res # сохраняем в адрес памяти sum число 470

nop (5)

.set

p1 8

sum 0b1111

**Примечание:**

1. Чтобы написать комментарий после инструкции следует поставить пробел и символ # а затем писать комментарий. Комментарий может находиться и на новой строке, он должен так же начинаться с символа #

Пример:

Add $1, $2, $3 # Складываем значения в регистрах

# Далее необходимо повторить несколько раз операцию NOP

NOP (4)

1. Ассемблер не чувствителен к регистру. То есть add то же самое, что и ADD, $a то же самое, что и $A, pup то же самое, что и PUP.
2. Необходимо строгое соблюдение пробельных символов внутри инструкции.