# *FAQ по асму в никсах от мыщъх'а*
# *(фрагмент)*

## *Q3:что такое Intel и AT&T нотация?*

AT&T синтаксис разрабатывался компанией AT&T в те далекие времена, когда никакого Intel'a вообще не существовало, процессоры менялись как перчатки и знание нескольких ассемблеров было вполне нормальным явлением. По сравнению с синтаксисом Intel, AT&T-синтаксис намного более избыточен, но это сделано умышленно с целью сокращения ошибок (хинт: на одном процессоре команда MOV может перемещать 32-бита, на другом 16, а на третьем вообще 64).

Отличия синтаксиса AT&T от Intel следующие:

- **имена регистров предваряются префиксом: "%":**
  - Intel: eax, ebx, dl;
  - AT&T: %eax, %ebx, %dl;
- **обратный порядок операндов: вначале источник, затем приёмник**:
  - Intel: mov eax, ebx;
  - AT&T: movl %ebx, %eax;
- **размер операнда задается суффиксом, замыкающим инструкцию, всего есть три типа суффиксов: "b" — байт (8-бит), "w" – слово (16-бит) и "l" – двойное слово (32-бита)**:
  - Intel: mov ah, al;
  - AT&T: movb %al, %ah;
  - Intel: mov bx, ax;
  - AT&T: movw %ax, %bx;
  - Intel: mov eax, ebx;
  - AT&T: movl %ebx, %eax;
- **числовые константы записываются в Си-соглашении**:
  - Intel: 69h;
  - AT&T: 0x69;
- **для получения смещения метки используется префикс "$", отсутствие которого приводит к чтению содержимого ячейки:**
  - Intel: mov eax, offset label;
  - AT&T: movl $label, %eax;
  - Intel: mov eax, [label];
  - AT&T: movl label, %eax;
- **в тех случаях, когда метка является адресом перехода, префикс "$" опускается:**
  - Intel: jmp label;
  - AT&T: jmp label;
  - Intel: jmp –;
  - AT&T jmp 0x69;
- **для косвенного перехода по адресу используется префикс "*":**
  - Intel: jmp dword ptr ds:[69h];
  - AT&T: jmp *0x69;
  - Intel: jmp dword ptr ds:[label];
  - AT&T: jmp *label;
  - Intel: jmp eax;
  - AT&T: jmp *%eax;
  - Intel: jmp dword ptr ds: [eax];
  - AT&T: jmp *(%eax);
- **использование префикса "$" перед константой используется для получения ее значения. знак (если он есть) ставится после префикса. константа без указателя трактуется как указатель:**
  - Intel: mov eax, 69h;
  - AT&T movl $0x69, %eax;

- Intel: mov eax, -69h;
- AT&T movl $-0x69, %eax;
- Intel: mov eax, [69h];
- AT&T movl 0x69, %eax

- **для реализации косвенной адресации базовый регистр заключается в круглые скобки, перед которыми может присутствовать индекс, записанный в виде числовой константы или метки _без_ префикса "$":**
  - Intel: mov eax, [ebx];
  - AT&T: movl (%ebx), %eax;
  - Intel: mov eax, [ebx+69h];
  - AT&T: movl 0x69(%ebx), %eax;
  - Intel: mov eax, [ebx+label];
  - AT&T: movl label(%ebx), %eax;
- **если регистров несколько, то они разделаются через запятую:**
  - Intel: mov eax, [ebx+ecx];
  - AT&T: movl (%ebx, %ecx), %eax;
- **для задания коэффициента масштабирования (scale) перед первым регистром ставится ведущая запятая (при использовании базово индексной адресации запятая опускается), а сам коэффициент отделяется другой запятой, без префикса "$":**
  - Intel: mov eax, [ebx*8];
  - AT&T movl (,%ebx, 8), %eax
  - Intel: mov eax, [ebx*8+label];
  - AT&T movl label(,%ebx, 8), %eax
  - Intel: mov eax, [ecx+ebx*8+label];
  - AT&T: movl label(%ecx, %ebx, 8);
  - Intel: mov eax, [ebx+ecx*8+label];
  - AT&T: movl label(%ebx, %ecx, 8);
- **сегментная адресация с использованием сегментных регистров отличается от Intel'а использованием круглых скобок вместо квадратных:**
  - Intel: mov eax, es:[ebx];
  - AT&T: mov %es:(%bx), %eax;
- **в командах переходов и вызовов функций непосредственные сегмент и смещение разделяется не двоеточием, а запятой:**
  - Intel: jmp far 10h:100000h (псевдоконструкция!)
  - jmp $0x10, $0x100000;