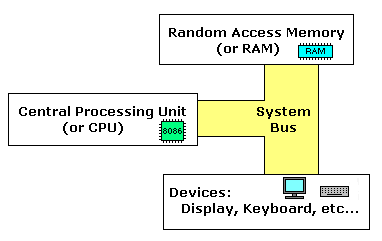
**Що таке Асемблер?**

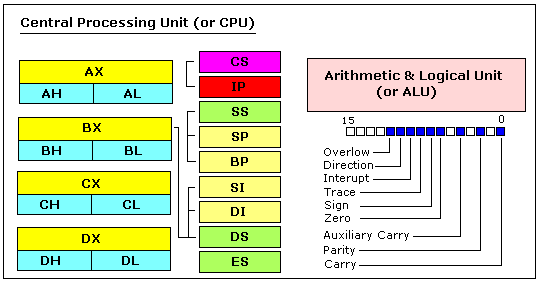
Мова Асемблера - це мова програмування низького рівня. Для початку ви повинні ознайомитися з загальною структурою комп'ютера, щоб в подальшому розуміти, про що йде мова. Спрощена модель комп'ютера:

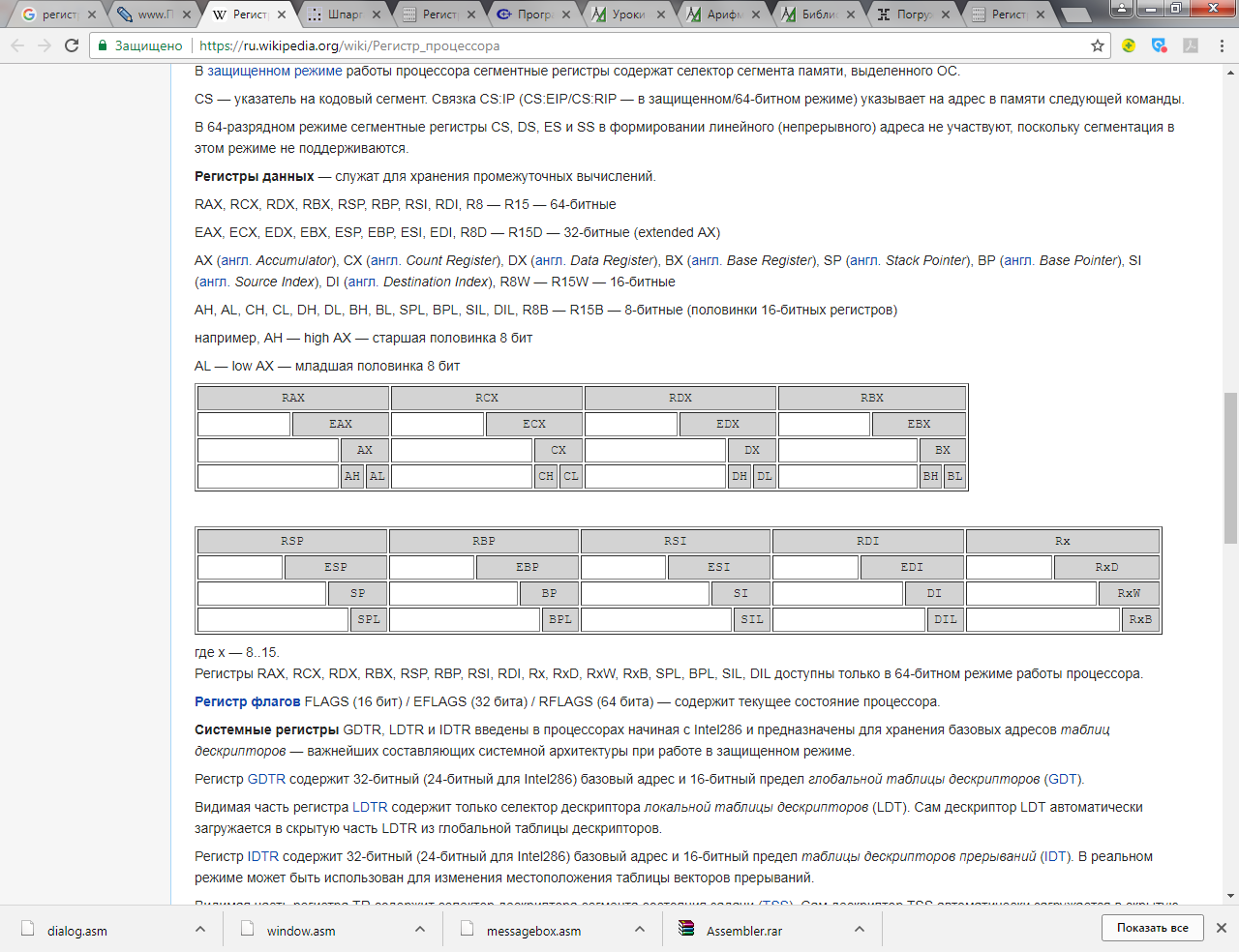
  
  
  
**System bus** - системна шина (забарвлена ​​жовтим кольором) з'єднує різні компоненти комп'ютера.

**CPU** - центральний процесор - це серце комп'ютера. Більшість обчислень відбувається в CPU.

**RAM** - оперативна пам'ять (ОЗУ). В оперативну пам'ять завантажуються програми для виконання.

**Будова процесора**



****

**РЕГІСТРИ ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ**

Процесор 8086 має 8 регістрів загального призначення, кожен регістр має ім'я:

• AX - регістр-акумулятор (розділений на два регістри: AH і AL).

• BX - регістр базової адреси (розділяється на BH / BL).

• CX - регістр-лічильник (розділяється на CH / CL).

• DX - регістр даних (розділяється на DH / DL).

• SI - регістр - індекс джерела.

• DI - регістр - індекс призначення.

• BP - покажчик бази.

• SP - покажчик стека.

Незважаючи на ім'я регістра, програміст сам визначає, для яких цілей використовувати регістри загального призначення. Основне призначення регістра - зберігання числа (змінної). Розрядність вищеописаних регістрів 16 біт, тобто, наприклад, 0011000000111001b (в двійковій системі) або 12345 в десяткового (людської) системі.

4 регістра загального призначення (AX, BX, CX, DX) розділені на дві частини. До кожної частини можна звертатися як до окремого регістру. Наприклад, якщо AX = 0011000000111001b, то AH = 00110000b, а AL = 00111001b. Старший байт позначається буквою "H", а молодший байт - буквою "L".

Оскільки регістри розташовані всередині процесора, то працюють вони значно швидше, ніж пам'ять. Звернення до пам'яті вимагає використання системної шини, а на це йде більше часу. Звернення до регістрів взагалі не забирає час. Тому ви повинні намагатися зберігати змінні в регістрах. Кількість регістрів дуже невелике і багато регістри мають спеціальне призначення, яке не дозволяє використовувати їх для зберігання змінних, але все ж вони є найкращим місцем для запису тимчасових даних і обчислень.

сегментні регістри

• CS - вказує на сегмент, що містить початкову адресу поточної програми.

• DS - зазвичай вказує на початковий адресу сегмента даних (змінних).

• ES - додатковий сегментний регістр.

• SS - містить початкову адресу сегмента стека.

Хоча в сегментних регістрах можна зберігати будь-які дані, робити це нерозумно. Сегментні регістри мають строго певне призначення - забезпечення доступу до блоків пам'яті.

Сегментні регістри працюють спільно з регістрами загального призначення для доступу до пам'яті. Наприклад, якщо ми хочемо отримати доступ до пам'яті з фізичною адресою 12345h (в шестнадцатиричном обчисленні), ми повинні встановити DS = 1230h і SI = 0045h. І це правильно, тому що таким чином ми можемо отримати доступ до пам'яті, фізичну адресу якої більше, ніж значення, яке може поміститися в одиночному регістрі.

Процесор обчислює фізичну адресу, множачи значення сегментного регістра на 10h і додаючи до отриманого результату значення регістра загального призначення (1230h \* 10h + 45h = 12345h):

Эффективный адрес  
  
Адреса, сформований за допомогою двох регістрів, називається реальним адресою.

За замовчуванням регістри BX, SI і DI працюють з сегментним регістром DS; регістри BP та SP працюють з SS.

Інші регістри загального призначення не можуть формувати реальну адресу!

Також, хоча BX може формувати реальну адресу, BH і BL не можуть!

РЕГІСТРИ СПЕЦІАЛЬНОГО ПРИЗНАЧЕННЯ

• IP - командний покажчик.

• Прапоровий регістр - визначає поточний стан процесора.

Регістр IP завжди працює спільно з сегментним регістром CS і вказує на виконувану в даний момент команду.

Прапоровий регістр автоматично змінюється процесором після математичних операцій. Він дозволяє визначати тип результату і передавати управління іншим ділянкам програми.

Взагалі ви не можете безпосередньо звертатися до цих регістрів.

Для доступу до пам'яті можна використовувати наступні чотири регістри: BX, SI, DI, BP.

Комбінуючи ці регістри всередині квадратних дужок [], ми можемо отримати доступ до різних розташувань в пам'яті.