

Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
"Архітектура комп'ютерів"
для студентів спеціальності
123 "Комп'ютерна інженерія"
всіх форм навчання
Частина 1

2017

Методичні вказівки до виконання лабораторних робіт з дисципліни "Архітектура комп'ютерів" для студентів спеціальності 123 "Комп'ютерна інженерія" всіх форм навчання. Частина 1 / Укл. Скрупський С.Ю. – Запоріжжя: ЗНТУ, 2017. – 46 с.

Укладачі: С.Ю. Скрупський, к.т.н., доцент

Рецензент: В.В. Шкарупило, к.т.н., доцент

Відповідальний за випуск: С.Ю. Скрупський, к.т.н., доцент

Затверджено:
на засіданні кафедри
"Комп'ютерні системи та
мережі"
Протокол № 2 від 13.09.2017

Рекомендовано до видання
НМК ФКНТ
Протокол № 2 від 28.09.2017

ЗМІСТ

1 ЛАБОРАТОРНА РОБОТА № 1. СТРУКТУРА КОМП'ЮТЕРА	4
2 ЛАБОРАТОРНА РОБОТА № 2. СТРУКТУРА І РОБОТА ЦЕНТРАЛЬНОГО ПРОЦЕСОРА.....	16
3 ЛАБОРАТОРНА РОБОТА № 3. РЕЖИМИ АДРЕСАЦІЇ ДАНИХ	20
4 ЛАБОРАТОРНА РОБОТА № 4. КОНТРОЛЕР КЛАВІАТУРИ. КОНТРОЛЕР ПРИНТЕРА	23
5 ЛАБОРАТОРНА РОБОТА № 5. ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ САПР АСТІВЕ-HDL	26
6 ЛАБОРАТОРНА РОБОТА № 6. ПРОГРАМУВАННЯ ПРОЦЕСОРА І8086 МОВОЮ АСЕМБЛЕРА.....	42
ЛІТЕРАТУРА	46

1 ЛАБОРАТОРНА РОБОТА № 1. СТРУКТУРА КОМП'ЮТЕРА

Мета роботи: знайомство зі структурою комп'ютера: цифрового, аналогового і гібридного, а також з їх основними функціональними блоками та принципами взаємодії.

1.1 Порядок виконання роботи

1.1.1 Вивчення структури цифрового комп'ютера (ЦК):

- ознайомитись із внутрішньою будовою цифрового комп'ютера;
- ознайомитись зі складом та призначенням компонентів системної плати;
- ознайомитись із засобами підключення різних периферійних пристроїв;
- за допомогою утиліт CPU-Z та Everest виписати основні технічні характеристики цифрового комп'ютера. Порівняти їх з технічними характеристиками Вашого домашнього комп'ютера.

1.1.2 Вивчення структури аналогового комп'ютера:

- ознайомитись з описом аналогового комп'ютера;
- ознайомитись зі структурою і різними блоками аналогового комп'ютера.

1.1.3 Вивчення структури гібридного комп'ютера:

- ознайомитись з описом гібридного комп'ютера, його основними системами та взаємодією цих систем;
- ознайомитись з основними функціональними блоками гібридного комп'ютера.

1.2 Зміст звіту

1.2.1 Призначення і структура цифрового комп'ютера.

1.2.2 Порівняння технічних характеристик Вашого домашнього комп'ютера та комп'ютера в аудиторії.

1.2.2 Призначення і структура персонального аналогового комп'ютера.

1.2.3 Призначення і структура гібридного комп'ютера.

1.2.4 Відповіді на контрольні питання.

1.3 Структура цифрового комп'ютера

У відповідності з формою відображення інформації ПК поділяються на аналогові АК – безперервної дії, цифрові ЦК – дискретної дії та гібридні ГК – змішаної дії.

Конструктивно ПК виконані у виді центрального системного блоку, до якого за допомогою розніму підключаються зовнішні пристрої: додаткові пристрої пам'яті, клавіатура, дисплей, принтер і ін. (рис.1.2).

Системний блок звичайно містить у собі системну плату, блок живлення, накопичувачі на дисках, рознім для додаткових пристроїв і *плати розширення* з контролерами – адаптерами зовнішніх пристроїв.

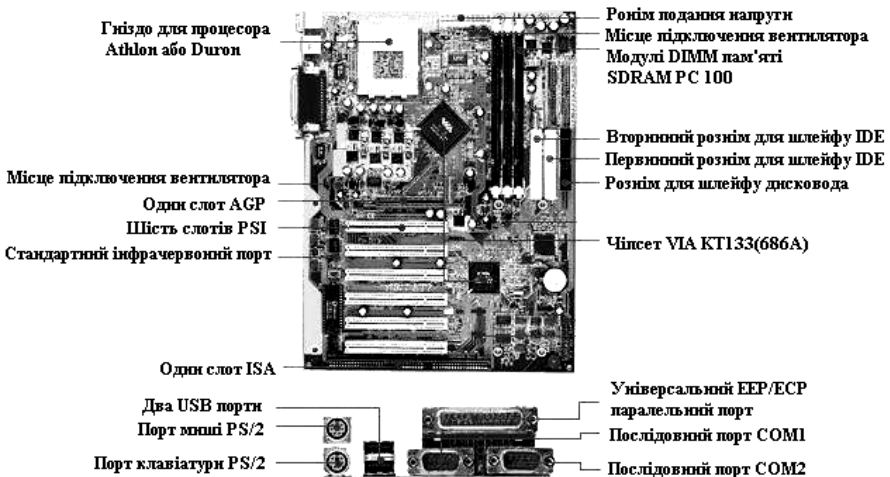


Рисунок 1.1 – Основні елементи та їх розташування на системній платі

На *системній платі* або *материнській платі* (рис.1.1), як правило, розміщуються:

- мікропроцесор;
- математичний співпроцесор;
- генератор тактових імпульсів;
- блоки (мікросхеми) ОЗП і ПЗП;
- адаптери клавіатури, НЖМД і НГМД
- контролер переривань;
- таймер і ін.

Мікропроцесор (МП). Це центральний блок ПК, призначений для керування роботою всіх блоків машини і для виконання арифметичних і логічних операцій над інформацією.

Генератор тактових імпульсів. Він генерує послідовність електричних імпульсів; частота імпульсів, що генеруються, визначає тактову частоту машини.

Системна шина. Це основна інтерфейсна система комп'ютера, що забезпечує сполучення і зв'язок усіх його пристроїв між собою. Системна шина забезпечує три напрямки передачі інформації:

- між мікропроцесором і основною пам'яттю;
- між мікропроцесором і портами вводу-виводу зовнішніх пристроїв;
- між основною пам'яттю і портами вводу-виводу зовнішніх пристроїв (у режимі прямого доступу до пам'яті).

Основна пам'ять. Вона призначена для збереження й оперативного обміну інформацією з іншими блоками машини і містить два види запам'ятовуючих пристроїв: постійний запам'ятовуючий пристрій (ПЗП) і оперативний запам'ятовуючий пристрій (ОЗП).

ПЗП служить для збереження незмінної (постійної) програмної і довідкової інформації, дозволяє оперативно тільки зчитувати інформацію, що зберігається в ньому, змінити інформацію в ПЗП неможливо.

ОЗП призначено для оперативного запису, збереження і зчитування інформації (програм і даних), що безпосередньо приймають участь у інформаційно-обчислювальному процесі, що виконується в ПК у поточний період часу.

Зовнішня пам'ять. Вона відноситься до зовнішніх пристроїв ПК і використовується для довгострокового збереження будь-якої інформації. Зовнішня пам'ять містить різноманітні види запам'ятовуючих пристроїв, але найбільш розповсюдженими є накопичувачі на жорстких (НЖМД) і гнучких (НГМД) магнітних дисках, оптичних дисках та магнітних стрічках.

Джерело живлення. Це блок, що містить системи автономного і мережного енергоживлення ПК.

Таймер. Це внутрішній електронний годинник, що забезпечує при необхідності автоматичне фіксування поточного

моменту часу (рік, місяць, годинник, хвилини, секунди і частки секунд). Таймер підключається до автономного джерела живлення – акумулятора і при відключенні машини від мережі продовжує працювати.

Зовнішні пристрої (ЗП) досить різноманітні і можуть бути класифіковані по ряду ознак. Так, за призначенням можна виділити наступні види ЗП:

- зовнішні запам'ятовуючі пристрої (ЗЗП) або зовнішня пам'ять ПК;
- пристрої вводу інформації;
- пристрої виводу інформації;
- засоби зв'язку і телекомунікації.

Відеомонітор (дисплей) - *пристрій* для відображення інформації, що вводиться або виводиться із ПК.

Пристрої мовного вводу-виводу відносяться до засобів мультимедіа, що швидко поширюються. Пристрої мовного вводу – це різні мікрофонні акустичні системи.

Пристрої мовного виводу – це різні синтезатори звуку, що виконують перетворення цифрових кодів у букви і слова, відтворені через гучномовці (динаміки) або звукові колонки, приєднані до комп'ютера.

До пристроїв вводу інформації відносяться:

- *клавіатура* – пристрій для ручного вводу числової, текстової і керуючої інформації в ПК;
- *графічні планшети (діджитайзери)* - для ручного вводу графічної інформації, зображень шляхом переміщення по планшету спеціального покажчика (пера); при переміщенні пера автоматично виконується зчитування координат його місця розташування і вводу цих координат у ПК;
- *сканери* (читаючі автомати) – для автоматичного зчитування з паперових носіїв і вводу в ПК машинописних текстів, графіків, малюнків, креслень.
- *маніпулятори* (пристрої вказівки): джойстик, миша, трекбол та ін. – для вводу графічної інформації на екран дисплея шляхом керування рухом курсору по екрану з наступним кодуванням координат курсору і вводом їх у ПК;

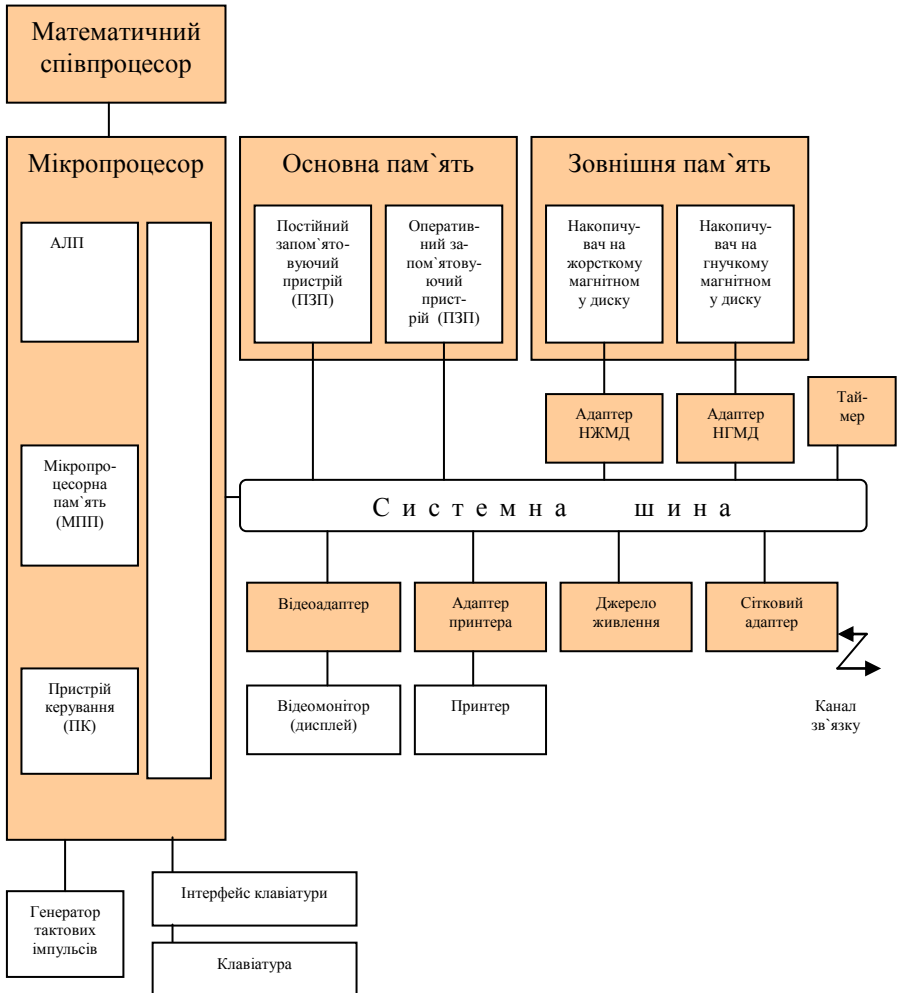


Рисунок 1.2 – Структурна схема персонального комп'ютера

– *сенсорні екрани* – для вводу окремих елементів зображення, програм або команд із поліекрана дисплея в ПК.

До пристроїв виводу інформації відносяться:

– *принтери* - друкуючі пристрої для реєстрації інформації на паперовий носій ;

– *графобудівники (плоттери)* – для виводу графічної інформації (графіків, креслень, малюнків) із ПК на паперовий носій; плоттери бувають векторні з кресленням зображення за допомогою пера і растрові: термографічні, електростатичні, стуйні і лазерні.

Пристрої зв'язку і телекомунікації використовуються для зв'язку з приладами й іншими засобами автоматизації (узгоджувачі інтерфейсів, адаптери, цифро-аналогові й аналого-цифрові перетворювачі і т.п.) і для підключення ПК до каналів зв'язку, до інших ПК і обчислювальних мереж (мережні інтерфейсні плати, мультіплексори передачі даних, модеми).

Засоби мультимедіа – це комплекс апаратних і програмних засобів, що дозволяють людині спілкуватися з комп'ютером, використовуючи різноманітні, природні для себе середовища: звук, відео, графіку, тексти, анімацію й ін.

Додаткові схеми. До системної шини і до МП ПК поряд з типовими зовнішніми пристроями можуть бути підключені і деякі додаткові плати з інтегральними мікросхемами, що розширюють і поліпшують функціональні можливості мікропроцесора: математичний співпроцесор, контролер прямого доступу до пам'яті, співпроцесор вводу-виводу, контролер переривань і ін.

Математичний співпроцесор широко використовується для прискореного виконання операцій над двійковими числами з плаваючою комою, над двійково-кодованими десятковими числами, для обчислення деяких трансцендентних, у тому числі тригонометричних функцій

Контролер прямого доступу до пам'яті звільняє МП від прямого керування накопичувачами на магнітних дисках, що істотно підвищує ефективну швидкість ПК.

Співпроцесор вводу-виводу за рахунок паралельної роботи з МП значно прискорює виконання процедур вводу-виводу при обслуговуванні декількох зовнішніх пристроїв (дисплей, принтер, НЖМД НГМД і ін.); звільняє МП від обробки процедур вводу-виводу, у тому числі реалізує і режим прямого доступу до пам'яті.

Для підключення периферійних пристроїв до ПК використовуються 2 типи портів: послідовні та паралельні. Деякі ПК мають ці порти на системній платі, а інші використовують плату розширення. Паралельний порт отримує і відправляє дані побайтно, послідовний же порт обмінюється з ПК побайтно, а з зовнішніми

пристроями побітно. Ці перетворення в послідовному порту виконує спеціальна мікросхема.

1.4 Структура аналогових комп'ютерів

Відповідно до визначення задачу на АК вирішують у тому вигляді, у якому вона подана. Функціональні блоки машини повинні виконувати весь комплекс математичних операцій, що необхідні для побудови структури досліджуваних рівнянь. Наприклад, якщо задане диференціальне рівняння n -го порядку вигляду:

$$\frac{d^n \cdot u}{d \cdot t^n} + a_{n-1} \cdot \frac{d^{n-1} \cdot u}{d \cdot t^{n-1}} + f_1(u) \cdot \frac{d^{n-2} \cdot u}{d \cdot t^{n-2}} - a_{n-3}(t) \cdot \frac{d^{n-3} \cdot u}{d \cdot t^{n-3}} + \\ + f_2(u) \cdot a_{n-4}(u) \cdot \frac{d^{n-4} \cdot u}{d \cdot t^{n-4}} - \dots - a_1 \cdot u = F(t) \cdot y,$$

то до складу машини повинні входити функціональні блоки для відтворення операцій додавання, віднімання, множення змінної величини на постійну, одержання функції довільного аргументу $f_1(u)$, функції часового аргументу $F(t)$, змінного коефіцієнту від довільного аргументу $a_{n-4}(u)$ і часу $a_{n-3}(t)$, множення двох змінних величин і операцій інтегрування і диференціювання. Різноманітний склад математичних операцій визначає набір функціональних блоків, що беруть участь у розв'язанні задачі.

Аналогові обчислювальні машини загального призначення мають приблизно однаковий склад устаткування. Узагальнена структурна схема сучасної аналогової машини показана на рис. 1.3. Зі схеми очевидно, що АК складається з ряду систем та блоків.

Під *системою* розуміють сукупність технічних засобів, призначених для забезпечення нормального функціонування машини. Розрізняють *основні* системи (керування, установки постійних і змінних коефіцієнтів, комутації, контролю й індикації) та *допоміжні* системи, до яких можна віднести системи живлення, термостатизування і вентиляції. Під блоком АК розуміють частину АК, що має певне функціональне призначення. Розрізняють основні блоки (наприклад, блоки операційних підсилювачів і операційні блоки, за допомогою яких виконуються різноманітні математичні операції) та допоміжні (наприклад, блоки перевірки підсилювачів та ін.).

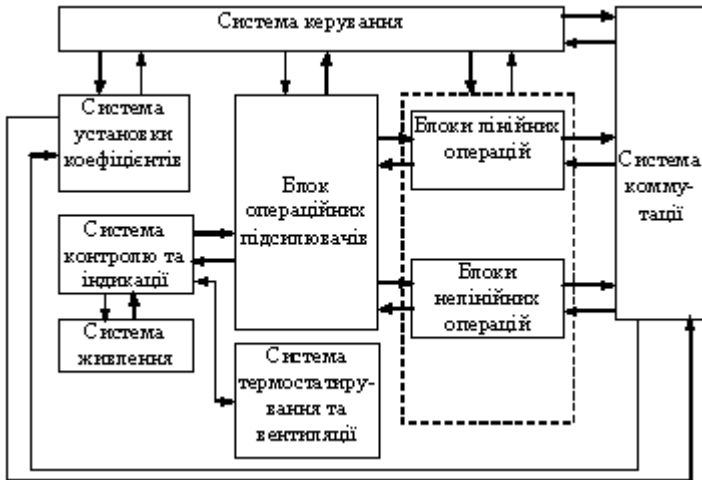


Рисунок 1.3 – Структурна схема АК

Блок операційних підсилювачів. Операційний підсилювач (ОП) - основний елемент сучасного АК. Він являє собою підсилювач постійного струму (ППС) із великим коефіцієнтом підсилення. На базі ОП у сполученні з пасивними елементами (резисторами, конденсаторами, діодами) виконують практично усі математичні операції в машині. Частина ОП використовується для допоміжних цілей, наприклад для отримання еталонних напруг. Кількість ОП визначається потужністю машини. Так, машина малої потужності МН-10 має 24 підсилювача, машина середньої потужності АВК-2(2)-64 підсилювача. У машинах великої потужності число ОП може досягати декількох сотень.

Операційні блоки. В залежності від математичної операції, що виконується їх поділяють на блоки лінійних і нелінійних операцій.

Блоки лінійних операцій. Ці блоки працюють разом із ОП та виконують операції інвертування, додавання, інтегрування, диференціювання, множення на постійний коефіцієнт. Вони містять набір прецизійних пасивних елементів - резисторів, конденсаторів і потенціометрів.

Блоки нелінійних операцій. Вони також працюють разом із ОП і призначені для перемножування двох співмножників, ділення, витягу квадратного кореня, відтворення функцій одного або двох довільних

аргументів і часу. Ці блоки складаються з пасивних лінійних та нелінійних елементів-діодів, резисторів, тиристорів і т.п. Кількість блоків може бути різним. У машинах малої потужності нелінійні блоки часто бувають змінними.

Таким чином, при розв'язанні задач на АК блоки лінійних та нелінійних операцій за допомогою системи комутації приводяться у взаємодію з блоком ОП. При цьому утворюються пристрої, що виконують ті або інші функції, або математичні операції. Тому такі пристрої часто називають *функціональними пристроями*.

Система керування АК. Ця система забезпечує взаємодію в часі усіх частин АК. Оператор, працюючий на машині, за допомогою пульта керування включає систему керування у роботу в одному із двох режимів: *підготовка* або *розв'язання задачі* (робота). У режимі *підготовки* відбувається налаштування блоків та їх комутація. У режимі *розв'язання задачі* здійснюється інтегрування з заданими початковими умовами і фіксація розрахунку. Розрахунок може відбуватися однократно або з автоматичним багатократним повторенням - *періодизацією*.

Система установки коефіцієнтів. Установка постійних та змінних коефіцієнтів відповідно до заданого рівняння - один із важливих етапів підготовки АК до розрахунку, що значною мірою впливають на точність результату. У машинах малої потужності постійні коефіцієнти звичайно встановлюють вручну. У машинах великої потужності для скорочення часу використовують автоматичну установку коефіцієнтів. Система установки змінних коефіцієнтів у АК малої потужності може бути відсутньою. У АК середньої і великої потужності змінні коефіцієнти вводяться різноманітними засобами, наприклад, за допомогою систем, що стежать.

Система контролю та індикації. Система призначена для контролю стану ОП та операційних блоків у різноманітних режимах роботи та спостереження за розв'язанням задачі. До складу її можуть входити стрілочні, цифрові, та друкуючі вольтметри. Для спостереження за розрахунком використовують багатопрменеві електронні індикатори. Розрахунок записується для подальшого аналізу та збереження на шлейфові осцилографи, самописи та графобудівники.

Система живлення. У задачу цієї системи входять перетворення напруги мережі в напруги, необхідні для роботи блоків

та стабілізація цих напруг. Зміна живлющих напруг сприймається в АК як зміна змінних, тому до стабільності напруг подаються підвищені вимоги.

Система комутації. Система забезпечує організацію необхідних з'єднань між окремими блоками АК за допомогою комутаційних шнурів. Шнури за допомогою вилок вставляють у гнізда на комутаційному полі та з'єднують блоки у відповідності зі схемою. У ряді випадків роблять декілька змінних комутаційних полів, які знімаються, що дозволяє робити комутацію окремо від машини. Цим досягається більш ефективне використання АК. У сучасних аналогових та аналого-цифрових комплексах керування комутаційною панеллю відбуваються автоматично по програмі, як, наприклад, в аналоговому обчислювальному комплексі АВК-32.

1.5 Структура ГК

Під *аналого-цифровим (гібридним) обчислювальним комплексом* або *системою* будемо розуміти сукупність аналогових, цифрових і аналого-цифрових машин та пристроїв, між якими в процесі розрахунку відбувається обмін інформацією. Вони можуть мати різноманітні схеми побудови в залежності від призначення:

а) для контролю та керування аналоговими об'єктами використовують цифровий керуючий процесор та пристрій перетворення і сполучення з об'єктом, що має необхідну кількість каналів аналого-цифрового та цифро-аналогового перетворення;

б) для моделювання аналогових об'єктів використовують цифровий керуючий процесор і сполучений з ним аналоговий процесор, потужність якого визначається складністю об'єкта, що моделюється.

в) такі ж обчислювальні засоби необхідні для створення проблемно-орієнтованих гібридних обчислювальних систем різноманітного призначення.

На рис. 1.4 показана структурна схема АЦОК. Його основу складають цифровий керуючий ЦКП та аналоговий АП процесори, пов'язані між собою інформаційними каналами. У схему входить ряд блоків, що забезпечують нормальне функціонування обох процесорів та зовнішніх пристроїв.

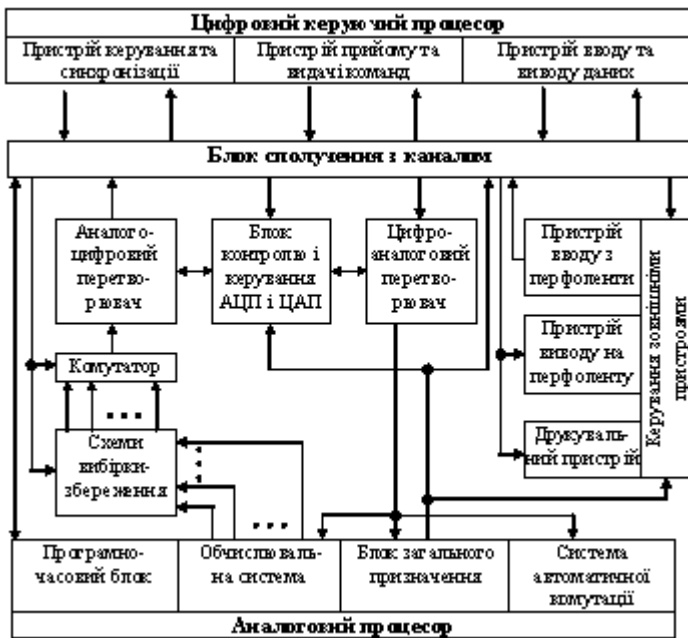


Рисунок 1.4 - Структурна схема аналого-цифрового комплексу

Цифрова частина АЦО являє собою ПК із розвинутою системою команд і системою переривання, що забезпечує оперативне керування аналоговою частиною, реальними об'єктами та різноманітними периферійними пристроями. Для зв'язку з аналоговою частиною повинні бути передбачені спеціальний режим роботи і відповідні команди і канали. Для виконання цієї задачі в систему керування ПК вводять блок зв'язку з аналоговою частиною.

Робота цифрової частини АЦО аналогічна роботі керуючої – у замкнутому контурі керування. Тому найбільш важливими характеристиками цифрового процесора є швидкість, що дозволяє працювати в реальному масштабі часу, обсяг оперативної пам'яті, засіб обміну даними з зовнішніми пристроями, можливість прямого доступу до пам'яті, робота зі стандартним каналом та ін.

Використання ПК великої потужності та доступ до зовнішніх пристроїв, що запам'ятовують, дозволяє будувати ГОС, що відрізняються високою продуктивністю.

Аналогова частина АЦОК повинна мати систему автоматичного керування операційними блоками (включаючи блоки комутації), здатну працювати під керуванням цифрових програм. Операційні блоки повинні мати підвищену точність виконання математичних операцій та широку смугу робочих частот, щоб забезпечити режим роботи швидкої періодизації рішення.

Аналоговий процесор забезпечує паралельне відтворення великої кількості математичних операцій і підключення зовнішніх пристроїв, а також безпосереднє підключення до стандартного каналу цифрового ПК.

Аналоговий процесор може працювати й в автономному режимі, при цьому для забезпечення власного програмного керування і рішення логічних задач у його склад вводяться елементи паралельної логіки (реєстри, лічильники, одновібратори) та елементи зв'язку аналогових і цифрових систем (компаратори, електронні ключі й ін.).

Сполучення цифрової й аналогової частин комплексу проводиться за допомогою АЦП і ЦАП. Якщо використовується один АЦП, то канали підключаються за допомогою пристрою вибірки-збереження та електронного комутатора. Для підвищення пропускної спроможності каналів аналого-цифрового перетворення можлива одночасна робота двох і більш АЦП.

У схемі звичайно є декілька ЦАП: лінійні (ЦАП) та перемножуючі ЦАП (ПЦАП). У процесі роботи здійснюється контроль за роботою АЦП і ЦАП за допомогою спеціальної схеми контролю по командах з аналогового або цифрового процесорів.

1.6 Контрольні питання

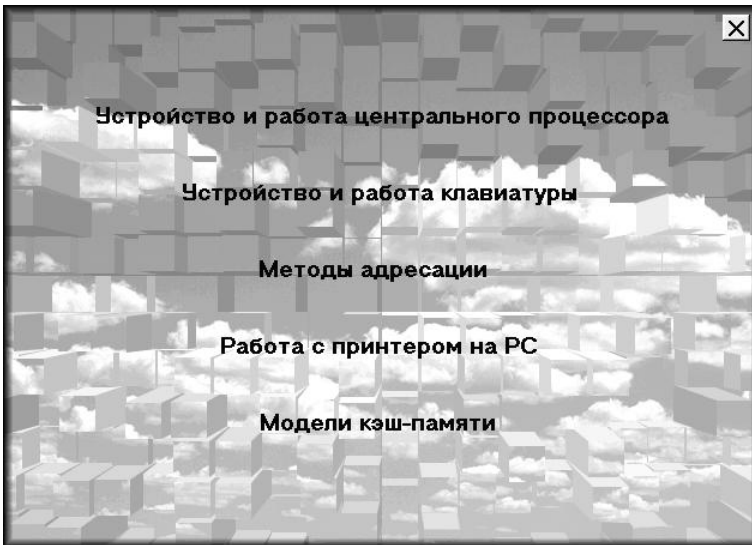
- 1.6.1 Області використання ЦК, АК, ГК.
- 1.6.2 Принципи дії ЦК, АК, ГК.
- 1.6.3 Основні компоненти ЦК, їх технічні характеристики.
- 1.6.4 Що таке шина?
- 1.6.5 Що таке архітектура комп'ютера?
- 1.6.6 Як порівняти продуктивність комп'ютерів різних архітектур?

2 ЛАБОРАТОРНА РОБОТА № 2. СТРУКТУРА І РОБОТА ЦЕНТРАЛЬНОГО ПРОЦЕСОРА

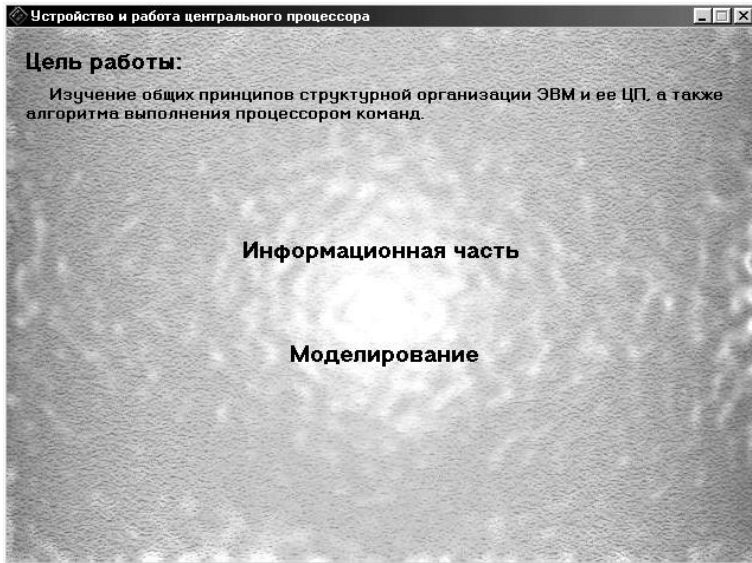
Мета роботи: вивчення структурної схеми цифрового комп'ютера та її складових частин, а також процесу виконання різноманітних команд.

2.1 Порядок виконання роботи

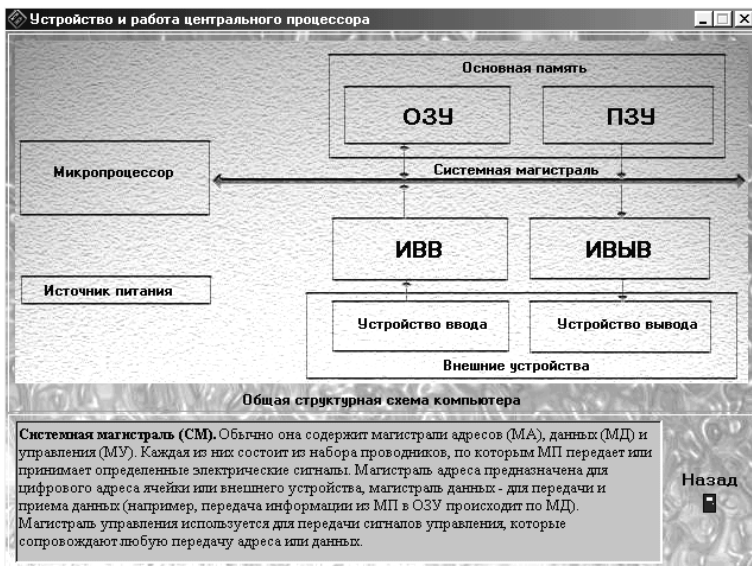
2.1.1 Виконати виклик системи “mmikurs” командою “\Commons\Comon2\Architecture\mmikurs.exe”. З’явиться вікно системи:



2.1.2 Вибрати з меню “Устройство и работа Центрального процессора”. З’явиться вікно:



2.1.3 Вибрати “Информационная часть”. З’явиться вікно:



У верхній частині вікна розташована структурна схема центрального процесора (ЦП), а у нижній – інформаційне вікно й кнопка “Назад”. В інформаційній частині наводиться короткий опис вибраних користувачем частин ЦП. Кнопка “Назад” використовується для повернення у попереднє вікно.

Вибираючи по черзі окремі блоки ПК та ЦП вивчити й законспектувати їх призначення. Зрисувати структурні схеми.

2.1.4 Повернутися у попереднє вікно й вибрати пункт “Моделирование”. З’явиться вікно, в якому відображені окремі функціональні блоки процесору та шини, що їх зв’язують й котрими в процесі моделювання будуть проходити керуючі сигнали, адреси та данні.



У нижній частині вікна є спеціальна секція, в якій наводиться опис кожного кроку (етапу) виконання команди. Праворуч вказується номер етапу, що виконується, та їх загальна кількість. Під ними дві стрілки: $\Rightarrow$ “шаг вперед”, $\Leftarrow$ “шаг назад”, що дозволяють або повернутися на попередній крок, або перейти до наступного. Там же знаходиться кнопка повернення до основного меню: “Назад”. Зліва від інформаційного вікна знаходиться піктограма “Команды”, натискання

на яку призводить до появи вікна з переліком команд. Під ним в спеціальному вікні з'являється призначення вибраної команди, а ще нижче розташована кнопка “Моделирование”, яка запускає вибрану команду на виконання. Праворуч знизу над стрілками “ $\Leftarrow \Rightarrow$ ” знаходиться регулятор сповільнення процесу моделювання етапів виконання команд. З початку переліку команд знаходиться команда “Нет”, після виконання якої вікно з командами зникає, тобто повертається початкове вікно пункту “Моделирование”.

2.1.5 Вивчити процес виконання п'яти різних команд.

2.1.6 Записати призначення всіх етапів виконання команд вказаних викладачем.

2.2 Зміст звіту

2.2.1 Мета роботи.

2.2.2 Опис структурної схеми ПК.

2.2.3 Призначення окремих блоків ПК та ЦП.

2.2.4 Опис команд вказаних викладачем та етапів їх виконання.

2.3 Контрольні питання

2.3.1 Наведіть структурні схеми ПК та ЦП.

2.3.2 Що таке біт, байт, слово, подвійне слово?

2.3.3 Назвіть типи команд за їх функціональним призначенням.

2.3.4 Наведіть приклади одно-, двох- та безадресних команд.

2.3.5 Чим відрізняються команди CISC та RISC процесорів?

2.3.6 Що таке конвеєризація команд?

2.3.7 Чим відрізняються елементи пам'яті ОЗП, ЗЗП та ПЗП?

2.3.8 Призначення регістрів ЦПК.

3 ЛАБОРАТОРНА РОБОТА № 3. РЕЖИМИ АДРЕСАЦІЇ ДАНИХ

Мета роботи: ознайомитися з основними режимами адресації даних, які використовуються у ПК, навчитися виконувати операції з адресами.

3.1 Порядок виконання роботи

3.1.1 Викликати систему “mmikurs” командою:
“\Commons\Comon2\Architecture\mmikurs.exe”

3.1.2 Вибрати в меню “Методы адресации”.

3.1.3 Вибрати “Информационная часть”. В інформаційній частині знаходиться короткий опис основних методів адресації і їх короткі характеристики. Законспектувати цю інформацію. Кнопка “Назад” використовується для повернення у попереднє вікно.

3.1.4 Кнопка “Дальше” відкриває вікно:



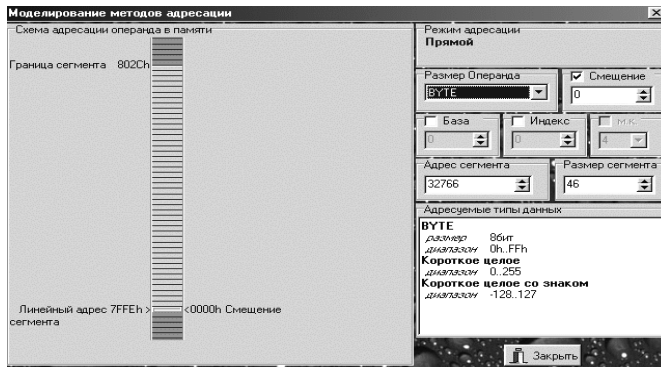
В лівій частині вікна знаходяться можливі методи адресації:

- прямий;
- регістровий;
- непрямої;
- базовий;
- індексний;

- індексний з масштабуванням;
- базовий – індексний;
- базовий – індексний з масштабуванням;
- базовий – індексний з зсувом;
- базовий – індексний з зсувом і масштабуванням.

В правій частині вікна знаходиться блок-схема формування лінійної адреси операнда при різних режимах адресації, а нижче описується, яким чином формується ця адреса та надається приклад цієї адресації на мові Асемблера. В лівій частині вікна внизу знаходяться 3 кнопки: „Моделирование”, „Назад”, „Выход”. Кнопка „Моделирование” демонструє формування на схемі сегмента пам’яті лінійної адреси операнда при вибраному методі адресації. За допомогою кнопки „Назад” виконується повернення до попереднього вікна. Натиснувши на „Выход” здійснюємо вихід з програми.

3.1.5 В полі “Режимы адресации” вибрати необхідний режим адресації. Після вибору режиму адресації за допомогою пункту “Моделирование” отримати моделювання даного режиму. З’явиться вікно:



В лівій частині цього вікна розташована схема пам’яті, де показані границі сегмента пам’яті, лінійна адреса операнда та його складові для вибраного метода адресації. В правій частині вікна зверху у відповідних віконцях задають: розмір операнда, зсув, адресу бази, значення індексу, масштабний коефіцієнт „м.к.”, адресу сегмента та його розмір.

Нижче наводиться розмір та діапазон значення типу даних, до якого відноситься вибраний операнд. Ще нижче є кнопка „Закрити”, яка вертає нас до попереднього вікна.

3.1.6 Розглянути та вивчити всі методи адресації, скласти звіт по даній роботі.

3.2 Зміст звіту

3.2.1 Мета роботи.

3.2.2 Зміст регістрів ЦП після кожної операції.

3.2.3 Формування адреси при різних режимах адресації.

3.2.4 Стислі відповіді на контрольні питання.

3.3 Контрольні питання

3.3.1 З якими основними режимами адресації працює центральний процесор?

3.3.2 Формування адреси при різних режимах адресації.

3.3.3 Як адресуватись до будь-якого елемента масиву слів, масиву байтів?

3.3.4 Як формується зсув? Які існують варіанти формування зсуву?

3.3.5 Режими адресації в RISC процесорах.

4 ЛАБОРАТОРНА РОБОТА № 4. КОНТРОЛЕР КЛАВІАТУРИ. КОНТРОЛЕР ПРИНТЕРА

Мета роботи: вивчити принципи роботи контролерів клавіатури та принтера. Засвоїти основні способи введення даних з клавіатури, виведення даних на твердий носій.

4.1 Порядок виконання роботи

4.1.1 Викликати систему “mmikurs” командою:
“\Commons\Comon2\Architecture\mmikurs.exe”

4.1.2 Вибрати в меню “Устройство и работа клавиатуры”.

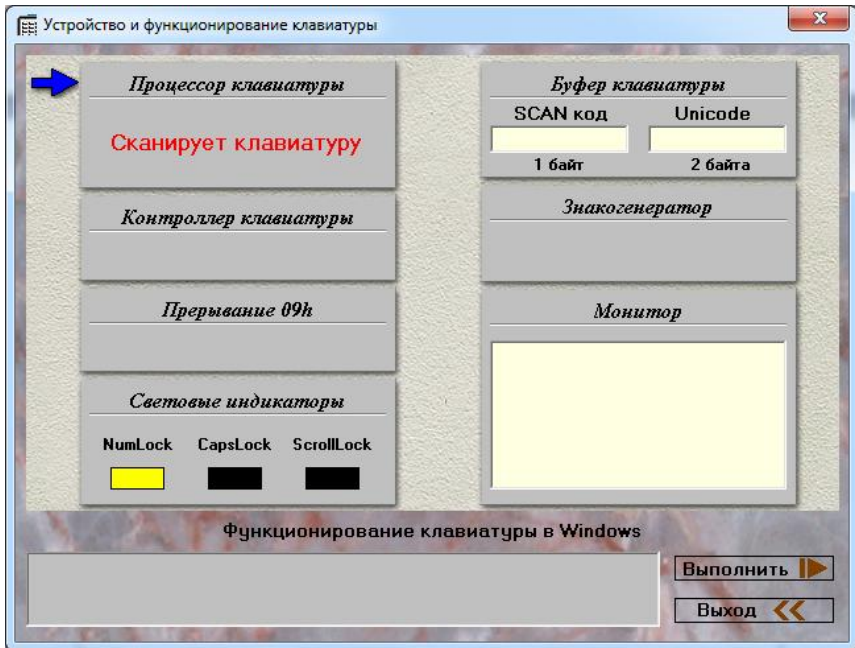
4.1.3 Вибрати “Информационная часть”. В інформаційній частині знаходиться короткий опис основних компонентів контролера клавіатури. Законспектувати цю інформацію. Кнопка “Назад” використовується для повернення у попереднє вікно.

4.1.4 Кнопка “Контроллер клавиатуры” відкриває вікно:



Натиснувши на кожний модуль на схемі можна отримати його опис.

4.1.5 Поверніться у вікно “Устройство и работа клавиатуры”. Виберіть пункт “Моделирование работы в Windows”. З'явиться вікно:

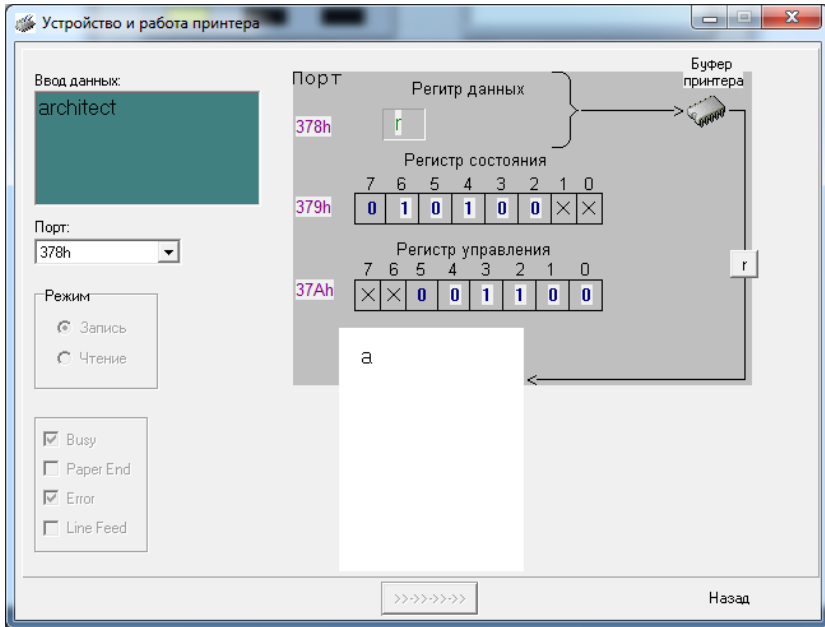


У вікні натисніть будь яку клавішу на клавіатурі та за допомогою кнопки "Выполнить" пройдіть по усіх етапах обробки клавіатурного натискання.

4.1.6 Поверніться до головного меню “mmikurs” та виберіть пункт "Работа с принтером на PC".

4.1.7 В інформаційній частині міститься опис регістрів принтера, що працює на паралельному порті. Законспекуйте його.

4.1.8 У вікні "Моделирование" виконайте моделювання роботи з принтером. Схематично замалюйте основні етапи передачі даних до принтера:



4.2 Зміст звіту

4.2.1 Мета роботи.

4.2.2 Сканування натискання клавіші на клавіатурі.

4.2.3 Основні етапи передачі даних до принтера.

4.2.4 Стислі відповіді на контрольні питання.

4.3 Контрольні питання

4.3.1 Яке призначення контролера клавіатури?

4.3.2 Що таке SCAN-код?

4.3.3 У чому відмінність кодування ANSI та UNICODE?

4.3.4 Інтерфейси підключення принтера, їх переваги та недоліки?

4.3.5 Які регістри використовуються при роботі з принтером на паралельному інтерфейсі?

5 ЛАБОРАТОРНА РОБОТА № 5. ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ САПР ACTIVE-HDL

Мета роботи: Вивчити структуру VHDL-проекту і функціональні можливості системи автоматизованого проектування Aldec Active-HDL.

5.1 Загальні відомості про Active-HDL

САПР Aldec Active-HDL є інтерактивним інтегрованим середовищем розробки і функціонально-структурного моделювання проектів цифрових систем (ЦС) на HBIC програмованої логіки (HBIC ПЛ). Її основними компонентами є компілятори мов опису апаратури і засоби функціонального моделювання проектів. Вбудовані додаткові засоби – текстовий редактор з функцією перевірки синтаксису, редактор кінцевих автоматів, бібліотека конструкцій мови, розвинута система допомоги та ін. – дозволяють значно скоротити час проектування і моделювання. Підтримуються мови опису апаратури *VHDL*, *Verilog* та *EDIF*. Можлива спільна робота Active-HDL і програмних засобів фірм-виробників HBIC ПЛ. Це дозволяє вести проектування цифрової системи від початку до кінця: від тексту програми до готових виробів в базисі FPGA і CPLD.

Користувачу доступні групи засобів розробки VHDL-проекту (рис. 5.1):

- засоби введення проекту (*Design Entry Tools*);
- засоби керування проектом (*Control Tools*);
- засоби виведення результатів моделювання (*Simulation Output Viewers*);
- засоби відлагодження проекту (*Debugging Tools*);
- ядро системи та інструментальні засоби.

До **засобів введення проекту** належать функціональні одиниці САПР Active-HDL: текстовий редактор з перевіркою, графічний редактор логічних схем, графічний редактор кінцевих автоматів, бібліотека конструкцій мови, конвертер HDL-тексту в логічні схеми.

Засоби керування проектом: менеджер проектів, менеджер файлів, ресурсів і структури проектів, менеджер керування напрямом проектування, менеджер бібліотек, вікно команд і повідомлень.

Функціональні одиниці, що є **засобами виведення результатів моделювання**: редактор часових діаграм, вікно таблиці перемикань.

До **засобів відлагодження проекту** належать: ядро системи моделювання, вікно паралельних процесів, вікно сигналів користувача, стек викликів підпрограм, вікно графічного відображення обміну інформацією усередині проекту.

Ядро системи та **інструментальні засоби** Active-HDL складають: генератор VHDL–проектів, генератор Verilog–проектів, генератор документів у форматі EDIF, компілятор мови VHDL, компілятор мови Verilog, компілятор EDIF.

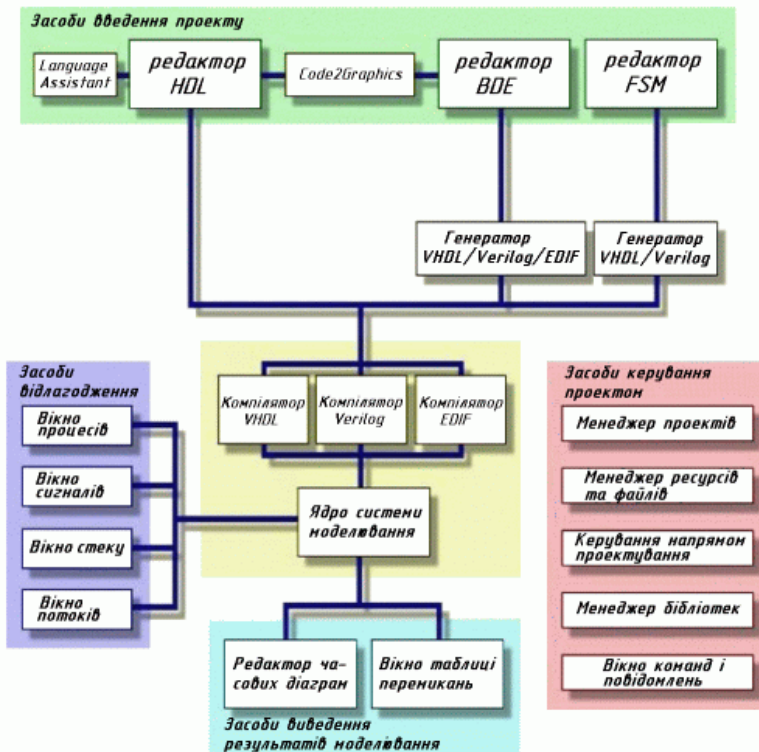


Рисунок 5.1 – Організація САПР Aldec Active-HDL

Часто буває достатньо скласти опис поведінки цифрової системи, алгоритму її роботи. З погляду автоматизації проектування

ЦС, якщо не визначений базис її реалізації, такий підхід є більш раціональним.

Будь-який об'єкт може мати декілька варіантів алгоритмічного опису архітектури. Різні за формою (синтаксисом), але єдині за значенням (семантикою) представлення одного і того ж алгоритму можуть задавати однакову поведінку об'єкту, але мати різну структурну реалізацію.

Можливість за допомогою **мови опису апаратури** (HDL) різними способами задавати один і той же алгоритм дозволяє за допомогою САПР Active-HDL автоматично синтезувати відмінні за складністю (рівнем деталізації) функціонально-структурні описи ЦС. Перевага, яку дає використання САПР Active-HDL, полягає в тому, що користувач, за допомогою конструкцій мови опису апаратури, сам указує рівень деталізації моделі ЦС.

Відповідно до цього складається **VHDL-проект** – формальний опис поведінки ЦС на мові VHDL. Користувач сам вибирає рівень деталізації опису цифрової системи (абстракції) з чотирьох можливих:

- **поведінковий рівень** (опис алгоритму роботи ЦС);
- **рівень регістрових передач** (рівень RTL);
- **функціонально-логічний рівень** (враховуються часові затримки розповсюдження сигналів);
- **рівень логічних елементів** (враховується час спрацювання логічних елементів).

VHDL-проект складається з **логічних файлів** опису компонентів ЦС, з'єднаних згідно формально визначеного закону функціонування ЦС. Логічні файли організовуються відповідно до ієрархії опису проектованої цифрової системи. Один з логічних файлів VHDL-проекту є файлом верхнього рівня ієрархії описів (*Top-level Design File*). Він грає роль організатора, що визначає архітектуру, склад ЦС і алгоритм її поведінки. Логічні файли нижніх рівнів ієрархії (*Low-level Design files*) містять детальний опис блоків і модулів ЦС. Ім'я проекту завжди співпадає з ім'ям логічного файлу верхнього рівня ієрархії проекту. До складу VHDL-проекту можуть бути залучені різноманітні допоміжні файли (*Ancillary Files*).

В САПР Active-HDL кожний проект зберігається в окремому каталозі, що зветься **каталогом проекту**. Проект і каталог, де він зберігається, мають однакові імена. Кожний каталог проекту містить

файл опису проекту (.ADF) і підкаталоги *SRC*, *GENERIC* і ін. Підкаталог *SRC* містить текстові файли мовою VHDL (.VHD), файли діаграм автоматів (.ASF) тощо. Підкаталог *GENERIC* містить робочі файли, які використовуються при компіляції і моделюванні.

Бібліотеки Active–HDL діляться на дві групи:

- **системні бібліотеки** поставляються разом з САПР Active–HDL і знаходяться в каталозі *VLIB*.

- **бібліотеки проектів** автоматично створюються окремо для кожного проекту і зберігаються в підкаталозі *GENERIC* проекту.

Кожна бібліотека складається з двох файлів: бібліотечного індексного файлу (розширення .INI) і головного бібліотечного файлу (розширення .MFG), що містить дані, що використовуються середовищем Active-HDL. В підкаталозі *VLIB* присутній також бібліотечний файл конфігурації *LIBRARY.CFG*, загальний для всіх бібліотек. Системні бібліотеки з каталогу *VLIB* наперед скомпільовані і готові до вживання у VHDL-програмах у вигляді *пакетів* (табл. 5.1).

Таблиця 5.1 – Вбудовані стандартні пакети САПР Active–HDL

Пакет	Призначення
STD.STANDARD	Стандартний пакет VHDL загального призначення (стандарт IEEE Std. 1076)
STD.TEXTIO	Стандартний пакет VHDL текстових операцій (стандарт IEEE Std. 1076)
IEEE.STD_LOGIC_1164	Підпрограми багаторівневої логіки (стандарт IEEE Std. 1164)
IEEE.VITAL_TIMING	Примітиви VITAL'95 (стандарт IEEE Std. 1076.4)
IEEE.VITAL_PRIMITIVES	Примітиви VITAL'95 (стандарт IEEE Std. 1076.4)
IEEE.STD_LOGIC_ARITH	Математичні підпрограми (стандарт IEEE Std. 1076.2)
IEEE.STD_LOGIC_SIGNED	Підпрограми IEEE (стандарт IEEE Std. 1076.5)
IEEE.STD_LOGIC_UNSIGNED	Підпрограми IEEE (стандарт IEEE Std. 1076.5)

Під час установки середовища Active–HDL пропонується підключити бібліотеки, що призначені для підтримки розробки проектів ЦС в базисі HBIC ПЛ ведучих виробників (Actel, Altera, Atmel, Cypress, Lattice, Lucent, Quicklogic, Synopsys і Xilinx).

Разом з середовищем поставляється докладна контекстна довідка та електронні підручники. Великий обсяг довідкових матеріалів зосереджено в навчальній системі Enhanced VHDL Tutorial (EVITA). Її можна викликати з САПР Active–HDL за допомогою пункту меню *Help* → *Interactive VHDL Tutorial*.

Розглянемо докладніше **процес розробки VHDL–проекту** нескладної ЦС в САПР Active–HDL. Наприклад, треба виконати функціональне моделювання поведінки демультимплексора DM(3), таблиця дійсності якого наведена в табл. 5.2. Нагадаємо, що демультимплексор – це цифровий пристрій, що перемикає вхідний сигнал W на один з виходів Y. На який саме – залежить від керуючих сигналів X.

Таблиця 5.2 – Таблиця дійсності демультимплексору DM(3)

Входи				Виходи							
w	x(2)	x(1)	x(0)	y(7)	y(6)	y(5)	y(4)	y(3)	y(2)	y(1)	y(0)
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

5.2 Створення нового проекту

5.2.1 Клацнувши на піктограмі на Робочому Столі Windows, запустить середовище Active–HDL.

5.2.2 На панелі інструментів Active–HDL клацніть піктограму *New Workspace* (створити нове робоче середовище).

5.2.3 З'явиться вікно **New Workspace**. В полі "*Type the workspace name*" введіть ім'я робочого середовища. Залиште активною

опцію "Add New Design to Workspace" (додати новий проект до робочого середовища). Клацніть **OK**.

5.2.4 Запускається майстер створення проекту **New Design Wizard**.

5.2.5 Налаштування параметрів проекту виконується за чотири кроки. Виберіть режим *Create an empty design* (створити порожній проект). Клацніть кнопку **Next**.

5.2.6 Друге вікно (другий крок) майстра призначено для вибору інструменту синтезу і імплементації. Необхідно також вказати тип HBIC ПЛІ, конфігурацію редактора BDE і мову опису проекту. В полі *Default HDL Language* виберіть **VHDL**.

5.2.7 Третій крок – визначення імені проекту, місця на диску для його зберігання і, якщо необхідно, налаштування імені бібліотеки проекту. За умовчанням ім'я проекту є ім'ям його бібліотеки. В полі *Type the design name* наберіть *Lab5* і клацніть кнопку **Next**.

5.2.8 Знов створеному проекту привласнено ім'я **Lab5**. В кінці роботи майстер створення проекту виводить інформацію про результати роботи. Файл проекту має ім'я *Lab5.adf*. Клацніть кнопку **Done**. На диску з'явиться каталог VHDL–проекту *Lab5*. Структура цього каталогу відображається у вікні **Design Browser**. Коли в проект *Lab5* будуть вноситись зміни щодо змісту файлів і директорій каталогу *Lab5*. Файл проекту *Lab5.adf* містить загальний опис проекту, компілятора і бібліотеки *Lab5.LIB*. У файл *console.log* будуть заноситися повідомлення про поточний стан проекту. Отримати доступ до повідомлень консолі можна у будь–який час натисненням на кнопку *Console*  на панелі інструментів або за допомогою комбінації клавіш <ALT+0>.

5.3 Створення нового документа (введення тексту програми)

5.3.1 Натисніть кнопку *Design Flow*  панелі інструментів (комбінація клавіш <Alt+3>). Відкриється вікно управління проектом (рис. 5.2).

5.3.2 Нам потрібен текстовий редактор. Для цього треба клацнути на піктограмі *HDE*. З'явиться вікно вибору мови опису апаратури. Виберіть **VHDL**.

5.3.3 З'явиться вікно **New Source File Wizard**. Цей майстер автоматично генерує частину тексту програми на мові VHDL – опис

портів в модулі *entity*. На першому кроці пропонується включити створюваний файл в поточний проект (*Add generated file to design*). Клацніть кнопку **Next**.

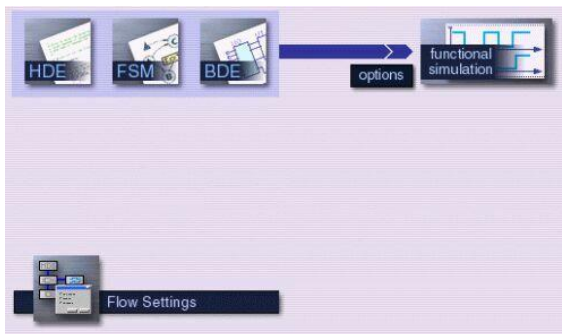


Рисунок 5.2 – Вікно Design Flow. Управління проектом

5.3.4 На другому кроці майстер пропонує дати імена VHDL-файлу, сутності і опису архітектури (*architecture*) проекту, що генерується. Текстовий файл тут названий *Lab5*, entity – *demux_3*, а опис архітектури має ім'я *beh* ("behavior" – поведінка). Для продовження клацніть **Next**.

5.3.5 Третій крок найвідповідальніший. Тут задаються порти, їх типи і режими роботи. Наш демультиплексор повинен мати один інформаційний вхід **w**, три входи керування **X** і вісім виходів **Y** (рис. 5.3).



Рисунок 5.3 – New Source File Wizard. Задавання портів

5.3.6 Список портів редагується за допомогою кнопок **New** і **Delete**. Для створення нового порту натисніть кнопку **New**. В полі *Name* введіть його ім'я. Для вибору напрямку порту використовується група параметрів *Port direction*. Для вибору типу порту треба клацнути кнопку **Type** і відкрити вікно **Port Type**. За умовчанням задаються типи *STD_LOGIS* або *STD_LOGIS_VECTOR*. В мові VHDL є чотири режиму роботи портів: **in** – вхідний порт, **out** – вихідний порт, **inout** – двонаправлений порт, **buffer** – порт-буфер. Параметр *Array Indexes* використовується для введення діапазонів масивів. Опис портів демультимплексору DM(3) має такий вигляд:

x: **in** STD_LOGIC_VECTOR (0 to 2);

w: **in** STD_LOGIC;

y: **out** STD_LOGIC_VECTOR (0 to 7);

5.3.7 На цьому робота з майстром шаблонів VHDL-програм **New Source File Wizard** закінчується. Клацніть **Done**. Буде створений шаблон VHDL-програми – моделі демультимплексора, в каталозі проекту з'явиться новий файл *Lab5.vhd* і відкриється вікно текстового редактора HDE.

Листинг 5.1 – Текст шаблону програми на мові VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity demux_3 is
    port(
        x : in BIT_VECTOR(0 to 2);
        w : in BIT;
        y : out BIT_VECTOR(0 to 7)
    );
end demux_3;
architecture beh of demux_3 is
begin
    -- enter your statements here --
end beh;
```

Шаблон VHDL-програми складається з трьох основних частин:

- перелік стандартних бібліотек, що використовуються (модуль *library*);
- повністю оформлений опис об'єкту (модуль *entity*);

- порожні операторні дужки опису архітектури (модуль architecture).

Модуль **library** визначає імена бібліотек проекту. VHDL-бібліотека може містити пакети, об'єкти, описи архітектури і конфігурації. В даному випадку використовується пакет *std_logic_1164* стандартної бібліотеки *IEEE*.

У модулі **entity** оголошується ім'я сутності (*demux_3*), описується інтерфейс між сутністю та середовищем, в якому вона функціонує. Тут визначаються типи і напрями сигналів.

Модуль **architecture** визначає внутрішню організацію, принципи функціонування сутності на мові опису апаратури VHDL.

5.3.8 В модулі `architecture` замість надпису — *enter your statements here* — необхідно додати код, який описує поведінку демультіплексора. В листингах 5.2 та 5.3 наведено два еквівалентні варіанти опису ЦС. Введіть програму, що наведено в листингу 5.2.

Листинг 5.2 – Перший варіант програми

[illegible]

```

        when "010" =>          y <= "00100000";
        when "110" =>          y <= "00010000";
        when "001" =>          y <= "00001000";
        when "101" =>          y <= "00000100";
        when "011" =>          y <= "00000010";
        when "111" =>          y <= "00000001";
    end case L2;
end if L1;
end process GO;
end beh;

```

Листинг 5.3 – Другий варіант програми

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity demux_3 is
    port(
        x : in STD_LOGIC_VECTOR(0 to 2);
        w : in STD_LOGIC;
        y : out STD_LOGIC_VECTOR(0 to 7)
    );
end demux_3;

architecture beh of demux_3 is
begin
    y(7) <= w and x(2) and x(1) and x(0) after 20 ns;
    y(6) <= w and x(2) and x(1) and not x(0) after 12 ns;
    y(5) <= w and x(2) and not x(1) and x(0) after 24 ns;
    y(4) <= w and x(2) and not x(1) and not x(0) after 18 ns;
    y(3) <= w and not x(2) and x(1) and x(0) after 15 ns;
    y(2) <= w and not x(2) and x(1) and not x(0) after 16 ns;
    y(1) <= w and not x(2) and not x(1) and x(0) after 21 ns;
    y(0) <= w and not x(2) and not x(1) and not x(0) after 14 ns;
end beh;
```

5.4 Компіляція проекту

Виконаємо компіляцію нашого проекту. Для цього можна натискувати клавішу <F11> або кнопку *Compile*  панелі інструментів. На рис. 5.4 показані кнопки запуску компіляції і результат компіляції VHDL-проекту. Якщо при складанні програми були допущені помилки, повідомлення про них можна побачити у вікні **Console**.

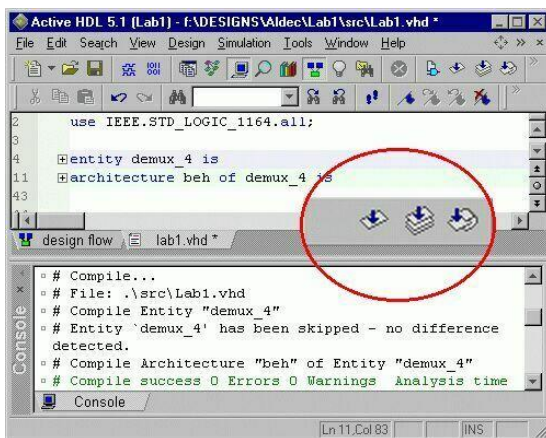


Рисунок 5.4 – Компіляція VHDL-проекту

5.5 Моделювання

Для того, щоб перевірити, чи правильно працює синтезований засобами САПР Active-HDL опис поведінки демультиплексора DM(3), виконаємо функціональне моделювання.

Перш ніж приступити до верифікації VHDL-проекту, необхідно скласти **тестову послідовність** вхідних сигналів w і $x[0..3]$. В найпростішому випадку тестова послідовність – це **вектор**, що містить всі можливі комбінації вхідних сигналів. Для демультиплексора DM(3) це $2^4=16$ векторів $[w, x_0, x_1, x_2, x_3]$.

5.5.1 Виберіть команду *File* → *New* → *waveform* або піктограму *New waveform* . З'явиться вікно редактора часових діаграм.

5.5.2 Додамо в редактор часових діаграм сигнали, що визначені в моделі `demux_3`. Клацніть правою кнопкою в лівій частині вікна

редактора. В контекстному меню виберіть команду *Add Signals*. З'явиться вікно **Add Signals** (рис. 5.5).

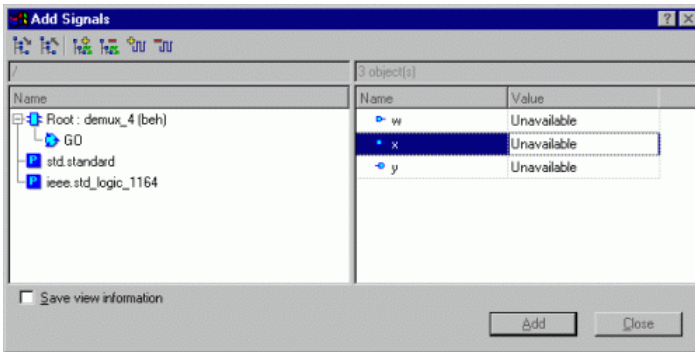


Рисунок 5.5 – Вікно Add Signals

5.5.3 Вибираючи по черзі сигнали **w**, **x**, **y** і клацаючи на кнопки **Add**, додайте ці сигнали у вікно редактора часових діаграм.

5.5.4 В головному меню Active-HDL виберіть команду *Simulation* → *Initialize Simulation*.

5.5.5 Тепер можна задати **стимулятори** моделювання. Щоб створити стимулятор треба виділити потрібний сигнал в редакторі часових діаграм, клацнути правою кнопкою і в контекстному меню вибрати команду **Stimulators**. Є декілька типів стимуляторів (таблиця 5.3).

5.5.6 Задамо стимулятор для сигналу **w**. Виберіть тип стимулятора **Hotkey**. Встановіть курсор в полі *Press new hotkey* і натисніть клавішу <w> клавіатури. Її будемо використовувати для перемикання рівня сигналу **w**.

5.5.7 Задамо стимулятор для вектору сигналів **x[0..2]**. Для перебору всіх комбінацій вектора сигналів зручно використовувати стимулятор типу **Counter**. Виділіть вектор **x** і встановіть тип стимулятора **Counter** (рис. 5.6).

5.5.8 Встановіть настройки лічильника відповідно до таблиці 5.4. Для того, щоб рахунок проводився в зворотному порядку, треба встановити прапорець *Reverse bits order*.

Таблиця 5.3 – Типи стимуляторів

 Value	Значення (Value). Дозволяє вибрати значення, що буде надане виділеному сигналу чи групі сигналів, об'єднаних у шину.
 Formula	Формула (Formula). Дозволяє задати часову діаграму за допомогою формули. Наприклад формула "0 0, 1 12" означає, що сигнал має значення '0' у момент часу 0 і змінюється на '1' після 12 пікосекунд.
 Hotkey	Гаряча клавіша (Hotkey). Дозволяє призначити сигналові гарячу клавішу, натискання якої призводитиме до зміни значення сигналу. За замовчуванням сигнал переключатиметься між 0 і 1, проте у цей список можна додати і інші допустимі для даного типу сигналу значення. При натисканні гарячої клавіші сигнал набуватиме наступного значення, включеного у список.
 Clock	Годинник (Clock). Дозволяє задати періодичний сигнал з визначеними параметрами (частотою, періодом тощо), користуючись зручним графічним редактором.
 Counter	Лічильник (Counter). Дозволяє моделювати сигнали на виходах лічильника як послідовність чисел в заданому напрямку лічби і в заданому базисі (двійковий код, код Грея та ін.).
 Predefined	Наперед визначений (Predefined). Це стимулятори синхросигналів або формульні, які мають власні імена, що дозволяє назначати їх декільком сигналам, не повторюючи кожен раз вводу параметрів.
 Custom	Стимулятор, що настроюється користувачем (Custom). Дозволяє вибрати наперед визначений стимулюючий сигнал з заданого переліку та поповнити цей перелік власними сигналами, що часто використовуються.

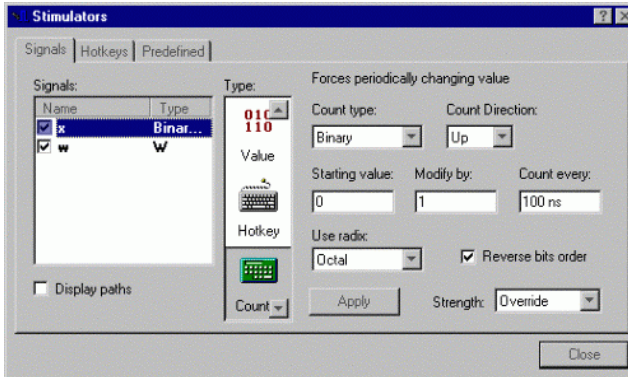


Рисунок 5.6 – Вікно Stimulators. Тип стимулятора – Counter

Таблиця 5.4 – Налаштування стимулятора для вектора $x[0..2]$

Налаштування		Значення
Count type	Тип лічильника	Binary
Count direction	Напрямок рахунку	Up
Starting value	Початкове значення	0
Modify	Величина зміни	1
Use radix	Підстава	Hexadecimal
Count every	Період рахунку	100 ns

5.5.9 Перевіримо, як веде себе модель *demux_3*, якщо сигнал **w** має високий логічний рівень. Натисніть клавішу <w> ("гарячу" клавішу, що зв'язана з сигналом **w**). Натискаючи клавішу <F5> вісім разів, виконайте покрокове моделювання роботи демультиплексора для першої половини тестової послідовності (коли сигнал **w**='1').

5.5.10 Знову натисніть клавішу <w> та виконайте покрокове моделювання демультиплексора для другої половини тестової послідовності (коли сигнал **w**='0').

5.5.11 Вибравши команду головного меню *Simulation* → *End Simulation*, зупиніть процес моделювання. На рис. 5.7 показаний результат функціонального моделювання правильно працюючого демультиплексора DM(3).

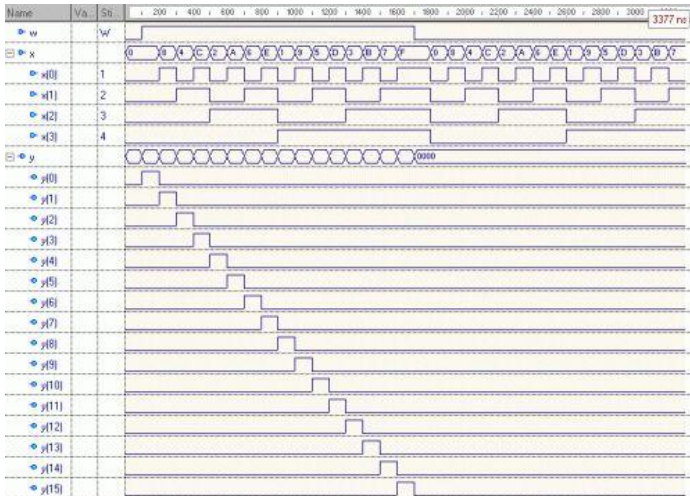


Рисунок 5.7 – Функціональне моделювання демультиплексора DM(3)

5.6 Порядок виконання роботи

– проект *Lab5*. Створіть і виконайте моделювання моделі *demux_3*. Для опису поведінки демультиплексора DM(3) використовуйте текст програми, наведений в лістингу 5.2. Текст програми та результати моделювання використовуйте для складання звіту.

– проект *Lab5-2*. Створіть новий проект і виконайте моделювання моделі *demux_3*, для опису якої використовується текст програми, наведений в лістингу 5.3. Текст програми та результати моделювання використовуйте для складання звіту.

– порівняйте результати моделювання і зробіть висновки.

– під час моделювання можна помітити, що сигнали перемикаються неодноразово. Цей ефект називається **гонкою сигналів**. Редактор часових діаграм має необхідні інструменти для докладного вивчення часових діаграм. Наприклад, кнопкою **Zoom in**  можна збільшити масштаб відображення, а натискання кнопки **Measurement Mode** , вмикає режим вимірювання часових інтервалів. На рис. 5.8 показаний приклад вимірювання часу встановлення сигналу.

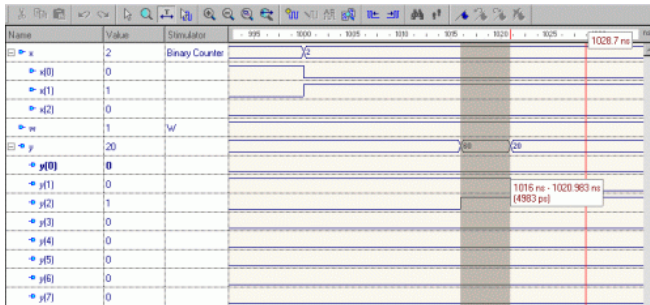


Рисунок 5.8 – Вимірювання часових інтервалів

– визначте значення максимальної затримки переключення сигналів вектору $y[0..7]$.

5.7 Зміст звіту

5.7.1 Мета роботи.

5.7.2 Таблиця істиності та логічна схема демультиплексора.

5.7.3 Тексти програм.

5.7.4 Результати моделювання.

5.7.5 Результати вимірювання часових інтервалів.

5.7.6 Висновки.

5.8 Контрольні питання

5.8.1 Організація VHDL-проекту цифрової системи.

5.8.2 Мови опису апаратури. Який зв'язок між ними і мовами програмування високого рівня загального користування.

5.8.3 Системи автоматизації проектування радіоелектронних пристроїв.

5.8.4 Організація САПР Active-HDL. Основні компоненти Active-HDL і їх призначення.

5.8.5 З яких частин складається опис об'єкту проектування?

5.8.6 Що таке тестова послідовність? Як задаються тестові набори вхідних сигналів? Наведіть приклади.

5.8.7 Типи стимуляторів. Як вони задаються?

6 ЛАБОРАТОРНА РОБОТА № 6. ПРОГРАМУВАННЯ ПРОЦЕСОРА I8086 МОВОЮ АСЕМБЛЕРА

Мета роботи: розвинути навички програмування мікроконтролерів мовою асемблера на прикладі i8086.

6.1 Порядок виконання роботи

6.1.1 Відкрити будь яке середовище IDE Assembler, наприклад, можна використати TASMIDE, що дозволяє створювати, налагоджувати та запускати проекти, написані мовою Асемблера.

6.1.2 Створити новий проект: "Файл -> Новый".

6.1.3 Ввести текст програми в основне вікно середовища. Для прикладу розглянемо програму, що виконує арифметичні дії над числами (лістинг 6.1).

Лістинг 6.1 – Проста програма мовою Асемблера

```
model    small      ; модель пам'яті
stack    256        ; розмір стека
.data    ; сегмент даних
val      dw 3        ; змінна розміром у слово (2 байти)
va2      dw 5
.code    ; сегмент коду програми
start:
    mov    bx,@data
    mov    ds,bx      ; фізичну адресу даних
                    ; занести в сегмент даних

    xor    bx,bx      ; очистити регістр bx
    add    bx,val     ; додати val до bx. Результат в bx
    sub    bx,va2     ; відняти va2 з bx. Результат в bx
    neg    bx         ; інвертувати знак регістра bx
    mov    ax,4c00h   ; стандартний вихід з програми
    int    21h
    end    start
```

6.1.4 Зберегти код програми у файлі, для цього виберіть пункт меню "Файл -> Сохранить". Ввести ім'я файлу та натиснути "Сохранить".

6.1.5 Для запуску проекту виберіть пункт меню "Проект -> Собрать и запустить", або натисніть F9. На екрані буде відображена консоль з результатами. Для програм, що виводять результати на консоль, перегляд результатів в такий спосіб є прийнятним, але для простих програм, що маніпулюють даними в пам'яті та в регістрах без виводу на консоль для перегляду результатів необхідно запускати відладник, наприклад Turbo Debugger.

6.1.6 Запустити відладник натиснувши на піктограмі "Запустить отладку". Після того відкриється відладник, в якому виберіть пункт меню "View -> CPU".

6.1.7 Для відладки програми потрібно натискати кнопку F7 (трасування) або F8 (перехід) та переглядати стан регістрів процесора i8086 після кожного кроку виконання коду (рис. 6.1).

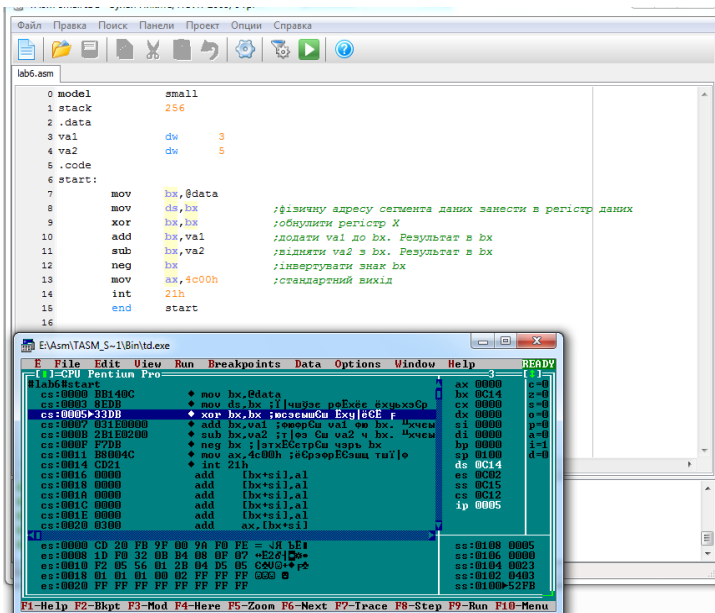


Рисунок 6.1 – Відладка програми у середовищі Turbo Debugger

6.1.8 Записати стан регістрів процесора після кожного кроку виконання програми.

6.1.9 Виконати завдання згідно варіанту:

Варіант 1: обчислити вираз: номер Вашого комп'ютера піднести до ступеня, що відповідає номеру дня місяця Вашого дня народження.

Варіант 2: обчислити вираз: кількість букв у Вашому прізвищі та по-батькові розділити на кількість букв в імені. Повторювати до тих пір, поки значення результату більше трьох.

6.2 Відомості про команди мовою Асемблера

Додавання та віднімання байт або слів здійснюється за допомогою команд ADD та SUB відповідно. Формат команд:

код команди операнд-приймач, операнд-джерело

Результат операції буде завантажений до операнда-приймача. Таким чином, наприклад, при складанні двох чисел (виконанні команди ADD), перше з чисел, які складаються, буде втрачене, а замість нього у відповідний регістр або у комірку пам'яті завантажуватиметься отриманий результат.

Множення та ділення чисельних даних у мові асемблер здійснюється за допомогою команд MUL та DIV для беззнакових чисел та за допомогою команд IMUL та IDIV для знакових чисел. В асемблері мікропроцесора Intel 8086 існують два формати команди множення - множення байта на байт та множення слова на слово. **При множенні байта на байт** один з множників знаходиться у регістрі AL, а другий - в байті пам'яті або в однобайтовому регістрі. Результат виконання операції множення буде розміщений в регістрі AX. **При множенні слова на слово** один з множників знаходиться у регістрі AX, а другий - в двох байтах пам'яті або в одному з регістрів процесора. Результатом множення в цьому випадку буде подвійне слово. Його молодші розряди будуть розташовані в регістрі AX, а старші - в регістрі DX. Треба мати на увазі, що ця команда ігнорує (затирає) будь-які дані, що містяться в регістрі DX. Команда ділення в асемблері також має два формати: ділення слова на байт та ділення подвійного слова на слово. **При діленні слова на байт** ділене міститься в регістрі AX, а дільник - у байті пам'яті або в однобайтовому регістрі. При діленні слова на байт результат ділення буде розміщений таким чином: частка - у регістрі AH, а остача - у регістрі AL. **При діленні подвійного слова на слово** ділене знаходиться в парі регістрів DX:AX, а дільник - у регістрі або в слові пам'яті. Після операції ділення остача завантажуватиметься в регістр DX, а частка в регістр AX. Формат команд множення/ділення:

код команди операнд-джерело

Команди умовного переходу забезпечують перехід в залежності від стану регістра ознак та забезпечують циклічність виконання коду. Джерелом умов переходів можуть бути будь-які команди, що змінюють значення арифметичних ознак, команди порівняння CMP, зміст регістру CX. Команда порівняння має синтаксис: *стр операнд_1, операнд_2*

вона дуже схожа на команду віднімання SUB, але результат віднімання не зберігається й не змінює значення операндів, а ознаки встановлюються відповідно до результату.

Типові команди умовного переходу, які виконуються після команди CMP:

JE адреса ; перехід на мітку якщо флаг ZF=0 (операнди рівні)

JL адреса ; перехід якщо знаковий операнд_1 < операнд_2

JB адреса ; перехід якщо беззнаковий операнд_1 < операнд_2

JG адреса ; перехід якщо знаковий операнд_1 > операнд_2

JA адреса ; перехід якщо беззнаковий операнд_1 > операнд_2

jcxz адреса_переходу ; перехід на мітку якщо регістр CX=0

Для організації **циклів** рекомендують використовувати команду LOOP адреса, що порівнює значення CX з нулем та переходить на мітку, якщо CX !=0. У CX необхідно попередньо записати значення кількості ітерацій.

6.3 Зміст звіту

6.3.1 Стан регістрів процесора після кожного кроку виконання програми, наведеної у п.6.1.

6.3.2 Код програми згідно варіанту та стан регістрів процесора після кожного кроку її виконання.

6.3.3 Відповіді на контрольні питання

6.4 Контрольні питання

6.4.1 Що таке регістр процесора, з яких складових частин він створюється?

6.4.2 Поясніть відмінність між синхронними та асинхронними регістрами.

6.4.3 Створіть цикл з п'яти ітерацій, в якому на кожній ітерації збільшіть значення регістра al на число, що відповідає номеру сьогоднішнього дня місяця.

6.4.4 Напишіть програму мовою асемблера, що підсумовує зміст трьох комірок пам'яті, результат розміщує у четвертій.

ЛІТЕРАТУРА

- 1 Таненбаум Э. Архитектура компьютера. Шестое издание / Э. Таненбаум, Т. Остин. – СПб.: Питер, 2013. – 816 с.
- 2 Мюллер С. Модернизация и ремонт ПК, 14-е издание / С. Мюллер. – М. : Вильямс, 2003. – 1184 с.
- 3 Смирнов А.Д. Архитектура вычислительных систем.- М.: Наука, 1990.- 320с.
- 4 Холленд Р. Микропроцессоры и операционные системы: Краткое справочное пособие: / Пер. с англ.-М.: Энергоатомиздат,1991.- 192с.
- 5 Семенец В.В. Проектирование цифровых систем с использованием языка VHDL: Уч. Пособие / В.В. Семенец, И.В. Хаханова, В.И. Хаханов.– Харьков: ХНУРЭ, 2003.– 492 с.
- 6 Бибило П.Н. Синтез логических схем с использованием языка VHDL.– М.: СОЛОН-Р, 2002.– 384 с.
- 7 Кондратенко Ю.П., Сидоренко С.А., Підпригора Д.М. Поведінковий синтез цифрових пристроїв у середовищі Active–HDL: Навчальний посібник. – Миколаїв: МФ НаУКМА, 2002.– 116 с.
- 8 Шапо Ф.С. Введение в VHDL – язык проектирования цифровых систем / Ф.С. Шапо, В.Ф. Шапо. – Одесса: Астропринт, 2001.– 220 с.
- 9 Архитектура комп'ютера [Електронний ресурс]. – Режим доступа: <http://inf1.info/computerarchitecture>