З метою підвищення рівня підготовки програмістів в області електроніки, ознайомлення з принципами побудови і функціонування процесорів на апаратному рівні, а також для формування уявлення про класичну архітектуру мікропроцесорів і мовами програмування низького рівня кафедрою електроніки та управляючих систем Харківського національного університету імені В.Н. Каразіна за участю студентів старших курсів була розроблена навчальна модель цифрового процесора. Вона побудована на мікросхемах малої ступеня інтеграції, дозволяє продемонструвати процес виконання програми на рівні окремих логічних елементів і оброблюваних ними сигналів [1].

Трохи пізніше на кафедрі був створений інтерфейсний модуль, який дозволив підключати макет процесора до персонального комп'ютера для моніторингу його стану і управління роботою в реальному часі. Завдяки розробці цього інтерфейсу і супутнього програмного забезпечення з'явилася можливість:

* завантажувати в оперативну пам'ять більш об'ємні програми, отримані шляхом трансляції з мови асемблер;
* відображати на екрані монітора поточний стан моделі процесора, вміст регістрів, пам'яті, стан шин;
* показувати хід виконання окремих команд;
* керувати роботою процесора як на рівні виконання окремих машинних циклів, так і на рівні виконання підпрограм.

Всі ці можливості істотно поліпшують наочність макета і дозволяють більш ефективно використовувати його в навчальному процесі.

Взаємодія моделі процесора з комп'ютером через створений інтерфейс здійснюється через паралельний восьмирозрядний LPT-порт [2]. Передача даних з комп'ютера в макет виконується по паралельній восьмирозрядній шині, а читання стану функціональних вузлів моделі проводиться напівбайтами за допомогою системи мультиплексорів і з використанням службових регістрів LPT-порту. При організації обміну, програма, безпосередньо, управляє роботою як регістрів даних, так і вхідних і вихідних службових регістрів в складі порту. Паралельна передача даних забезпечує високу швидкодію системи в цілому.

Модель цифрового процесора активно використовується в навчальному процесі на двох факультетах – комп’ютерних наук і фізико-технічному, Харківського національного університету імені Каразіна. Вона викликає величезний інтерес у студентів під час проведення лабораторних робіт і при активній участі ведеться її вдосконалення. Зокрема один студент розробив для даної моделі додатковий модуль оперативної пам’яті ємністю 32 кілобайт, яка істотно розширює можливості самої моделі.

Конструкція інтерфейсів регістрів (портів) дозволила на пряму підключати до моделі процесора рідкокристалічний індикатор і найпростішу клавіатуру, перетворивши модель на найпростішу модель комп’ютера.

Декілька років тому було розроблено програму, яка передбачає можливість читання і запису виконуваного коду, функції зберігання підготовлених програм на диску, яка дає можливість виконання, моніторингу і індикації стану процесора. Але функціональність бажала кращого. В неї було декілька принципових помилок, що час від часу викликали некоректну роботу програми, також вона застаріла для нових операційних систем. Але головною проблемою було те, що вона не давала можливість працювати з пам’яттю 256 байт, що значно звужувало можливості даної програми.

В своїй бакалаврській дипломній роботі я почала розробляти програму, яка буде відповідати всім вимогам, коректно працювати, з ергономічним дизайном, зрозумілим для користувача. В результаті було створено дизайн інтерфейсу і розроблено сам інтерфейс користувача, який протестований на різних операційних системах і комп’ютерах. Результат тестування показав, що створена інтерфейсна програма являється кросбраузерною.

Але не вирішена головна задача. Так як за допомогою мови JavaScript, на якій був розроблений інтерфейс користувача, на сьогоднішній день неможливо управляти апаратними ресурсами комп’ютера, то потрібно розробити ще одну програму, яка матиме таку можливість, а також буде кросс-платформною. Но зміна програмного забезпечення неможлива без розробки нового інтерфейсного модуля.

**2 Проблематика**

Робота з моделлю здійснювалася при використанні операційних систем, що надають можливість доступу до апаратних ресурсів комп'ютера (із серії операційних систем Windows робота була можлива з версіями Windows 98, Windows Vista, Windows XP).

Але на сьогоднішній день всі перераховані вище операційний системи не підтримуються компанією Майкрософт і поступово витісняються з використання. На зміну їм приходять більш нові версії, такі як Windows 7, Windows 8, Windows 10. У всіх цих системах з метою забезпечення безпеки та їх стійкості до шкідливих програм максимально обмежений доступ до апаратних ресурсів комп'ютера. Це робить неможливим використання раніше створеної програми з управління моделлю процесора, так як в існуючій версії передача даних здійснюється через службові регістри зовнішнього порту комп'ютера. Скоріш за все, наступні операційні системи будуть ще більше обмежувати можливості прямого управління портами без використання спеціально розроблених драйверів, які передбачають відповідні правила доступу і обмеження повноважень.

Сучасна тенденція розвитку персональних комп'ютерів визначена в першу чергу комерційними інтересами виробників і орієнтована на задоволення потреб найбільш широкого контингенту користувачів. Це, в першу чергу, - реалізація комунікативних, мультимедійних та ігрових функцій. З цієї причини в сучасних комп'ютерах «зникли» паралельні порти введення-виведення. Передача даних в основному здійснюється через USB-порти, а також через порти, в основу роботи яких покладено радіочастотний принцип передачі інформації (технології Bluetooth і Wi Fi). В результаті створене раніше обладнання неможливо підключити до сучасних ноутбуків і стаціонарних персональних комп'ютерів.

У зв'язку з цим виникла необхідність для подальшого успішного використання моделі процесора розробити новий модуль апаратного інтерфейсу і нове програмне забезпечення, яке дозволило б управляти апаратурою при роботі з сучасними комп'ютерами в операційних системах Windows 7 ... Windows 10.

**Мета роботи**

Метою дипломної роботи є дослідження та аналіз методів організації взаємодії в комп’ютерних системах управління та розробка на результатах цього аналізу нового апаратно-програмного комплексу для моніторингу та управління моделлю цифрового процесора.

**Галузь застосування**

Галузь застосування даного дослідження є учбовий процес з підготовки фахівців по програмуванню та електроніці на технічних факультетах Харківського національного університету імені Каразіна.

**Опис об’єкту та аналіз можливостей управління**

При розробці моделі процесору в її конструкції був передбачений роз’єднувач для підключення комп’ютерного інтерфейсу. На контакти цього роз’єднувача виведено:

* внутрішня шина даних процессора;
* шина адреси;
* сигнали управління оперативно-запам'ятовуючим пристроєм(ОЗП);
* вихідні шини регістрів арифметико-логічного АЛУ (25:09)
* шина командного слова
* Основні сигнали управління процесора
* Сигнали, які передають стан признакових бітів

Завдяки такій апаратній організації існує можливість повного управління роботою ОЗП: заносити до нього код програми зчитувати та очищати пам’ять. Окрім того є можливим безпосереднього управління усіма апартними процесами, що відбуваються в моделі. Комп. Має можливість зчитувати поточний стан усіх шин та основних регістрів процесору і за допомогою відповідного ПЗ (ссілка на бакалавр диплом) відображати на моніторі.

Контроль стану учбового обладнання та управління ним здійснюється через паралельні шини передачі даних саме тому існуючий інтерф модуль було побудовано на основі паралельного комп’ютерного інтерфейсу CENTRONIX, який нажаль на сьогодні не використовується в сучасних комп’ютерів.

Тому однією з основних задач даного дослідження є пошук схемних та програмних рішень, які могли б замінити існуюче обладнання на таке, яке зараз розповсюджене у сучасних комп’ютерах і яке буде використовуватись у майбутньому (можно описать и нарисовать старій интерфейсній модуль)

Досліждення можливостей управління до паралельних та послідовних портів комп’ютера. Вивчення існуючих програмних технологій для можливостей організацій такого прогрманого доступу. Єксперементальне дослідження різноманітних варіантів передачі данних до управляючої системи та аналіз отриманих результатів

**Вибір технологій для реалізації**

Так як згадувалось раніше, вже була створена програма, точніше інтерфейс користувача. Він був написаний за допомогою мов програмування HTML, CSS і JavaScript. Це гарний, відновлений інтерфейс, саме головне, він відповідає основній критерії – кросс-платформенності. Було проведено тестування на різних операційних системах і комп’ютерах – в результаті якого можна зробити висновок, що даний інтерфейс відображає данні коректно, а функції запису і зчитування інформації з файлу виконуються так як і очікується. Тому доцільно продовжувати роботу в даному напрямку, тобто розробляти веб-програму.

Але в попередній дипломній роботі я зіткнулася з проблемою, що за допомогою мови програмування JavaScript дуже важко програмувати для апаратної частини, тому що розробки в цій області тільки починають розвиватися і немає багато матеріалу для навчання, також треба мати спеціальні модулі, які зараз коштують недешево та нешироко розповсюдженні. Тому було вирішено розробити програму апаратного управління.

Дану програму треба розробити так, щоб вона могла спілкуватися і з вже існуючим інтерфейсом користувача і мала доступ до апаратних ресурсів комп’ютера. Із всіх існуючих на сьогоднішній день технологій найкращим варіантом є web-програма для локального застосування.

У результаті пошуку інформації в інтернеті на тему розробки web-програми для локального застосування можна сказати, що із-за того що переважна більшість відвідуваних web-сайтів доступно через інтернет, однак багато компаній прийшли висновку, що і розробка додатків для внутрішніх мереж займає важливе місце. Цю ідею можна розвинути далі і розробити повнофункціональні web-додатки, які ніколи не будуть відправляти пакети даних через мережевий інтерфейс. Також важливим фактом є те, що досвідчені web-розробники іноді витрачають великі зусилля на вивчення якого-небудь GUI-інструментарію, хоча те, що дійсно потрібно в їх ситуації - це простий CGI-скрипт.

У всякому разі, web-додаток, призначений для роботи тільки в локальній мережі, набагато простіше додатків для універсального застосування. Це дає можливість легко задати вимоги для браузера, а продуктивність сервера, швидше за все, не буде великою проблемою. Прості додатки, що використовують стандартні віджети CGI-форм і їм подібні, можна написати за набагато менший час, ніж треба було б на розробку самодостатніх додатків. Додатки, побудовані навколо обробки форм або даних, часто є відмінними кандидатами на реалізацію в якості звичайних web-сервісів.

В моєму випадку такий спеціальний додаток може забезпечити більш витончене і просте рішення поставленої задачі.

**Чим допоможе web-браузер?**

Якщо проаналізувати, що вміє web-браузер, чого не вміють інші програми? Відповідь очевидна: нічого. Але тоді що ж можна написати на високорівневих мовах, чого не можна виконати в машинних кодах? Знову ж таки, нічого. Перевага при використанні web-браузера в якості інтерфейсу полягає в тому, що все основне кодування вже виконано. Не потрібно відстежувати події зміни розмірів або розгортання вікна або події меню. Все, що потрібно зробити, це прочитати фрагмент даних з запиту і обробити його.

Web-браузер робить ще одну дуже корисну річ: він надає безліч налаштувань і переваг, про які вам не потрібно турбуватися. Користувач може змінювати розмір шрифту під час роботи. Аналогічно, якщо згенеровані вихідні дані представляються в простій і акуратній HTML-формі, їх можна легко і швидко вивести на друк. Багато функцій, які в іншому випадку довелося б реалізовувати самостійно (наприклад, збереження вихідних даних в файл, вивід вихідних даних на друк, зміна розмірів вікон), вже реалізовані. [https://www.ibm.com/developerworks/ru/library/wa-localwebsrv/index.html]

**Взаємодія інтерфейсу с програмою-сервером**

Перед тим як користувач побачить вміст сайту у себе на екрані, браузер робить запит на сервер щоб отримати вміст.

Перше що потрібно запам'ятати - браузер запитує і отримує дані за допомогою HTTP протоколу, тому його (браузер) ще називають HTTP клієнтом.

Протокол передачі Гіпертексту (HTTP - англ. HyperText Transfer Protocol) - спеціально розроблений протокол як основа World Wide Web і використовується для передачі всіх необхідних даних: HTML коду, зображень, CSS файлів, javascript і т.д. [http://xiper.net/learn/also-need-to-know/how-does-a-browser-communicate-with-the-server]

Відповідно до специфікації OSI, HTTP є протоколом прикладного (верхнього, 7-го) рівня. Актуальна на даний момент версія протоколу, HTTP 1.1, описана в специфікації RFC 2616.

Протокол HTTP припускає використання клієнт-серверної структури передачі даних. Клієнтську програму формує запит і відправляє його на сервер, після чого серверне програмне забезпечення обробляє цей запит, формує відповідь і передає його назад клієнтові. Після цього клієнтську програму може продовжити відправляти інші запити, які будуть оброблені аналогічним чином.

Завдання, яке традиційно вирішується за допомогою протоколу HTTP - обмін даними між призначеним для користувача додатком, що здійснює доступ до веб-ресурсів (зазвичай це веб-браузер) і веб-сервером. На даний момент саме завдяки протоколу HTTP забезпечується робота Всесвітньої павутини.

Також HTTP часто використовується як протокол передачі інформації для інших протоколів прикладного рівня, таких як SOAP, XML-RPC і WebDAV. У такому випадку говорять, що протокол HTTP використовується як «транспорт».

API багатьох програмних продуктів також має на увазі використання HTTP для передачі даних - самі дані при цьому можуть мати будь-який формат, наприклад, XML або JSON.

Як правило, передача даних по протоколу HTTP здійснюється через TCP / IP-з'єднання. Серверне програмне забезпечення при цьому зазвичай використовує TCP-порт 80 (і, якщо порт не вказано явно, то зазвичай клієнтське програмне забезпечення за замовчуванням використовує саме 80-й порт для відкритих HTTP-з'єднань), хоча може використовувати і будь-який інший.

Для того, щоб сформувати HTTP-запит, необхідно скласти стартовий рядок, а також задати принаймні один заголовок - це заголовок Host, який є обов'язковим, і повинен бути присутнім в кожному запиті. Справа в тому, що перетворення доменного імені в IP-адресу здійснюється на стороні клієнта, і, відповідно, коли ви відкриваєте TCP-з'єднання, то віддалений сервер не володіє жодною інформацією про те, яка саме адреса використовувалася для з'єднання: це могла бути, наприклад, адреса exzample.google.com, google.com або m.google.com - і у всіх цих випадках відповідь може відрізнятися. Однак фактично мережеве з'єднання у всіх випадках відкривається з вузлом 212.24.43.44, і навіть якщо спочатку при відкритті з'єднання було поставлено не ця IP-адреса, а будь-яке доменне ім'я, то сервер про це ніяк не дізнається - і саме тому цю адресу необхідно передати в заголовку Host.

Стартовий (початковий) рядок запиту для HTTP 1.1 складається за такою схемою:

Метод URI HTTP / Версія (1.1)

Наприклад (такий стартовий рядок може вказувати на те, що запитується головна сторінка сайту):

GET / HTTP / 1.1

Метод (в англомовній тематичній літературі використовується слово method, а також іноді слово verb - «дієслово») являє собою послідовність з будь-яких символів, крім керуючих і роздільних, і визначає операцію, яку потрібно здійснити з зазначеним ресурсом. Специфікація HTTP 1.1 не обмежує кількість різних методів, які можуть бути використані, проте в цілях відповідності загальним стандартам і збереження сумісності з максимально широким спектром програмного забезпечення як правило використовуються лише деякі, найбільш стандартні методи, зміст яких однозначно розкритий в специфікації протоколу.

URI (Uniform Resource Identifier, уніфікований ідентифікатор ресурсу) - шлях до конкретного ресурсу (наприклад, документа), над яким необхідно здійснити операцію (наприклад, в разі використання методу GET мається на увазі отримання ресурсу). Деякі запити можуть не ставитися до будь-якого ресурсу, в цьому випадку замість URI в стартовий рядок може бути додана зірочка (астеріск, символ «\*»). Наприклад, це може бути запит, який відноситься до самого веб-сервера, а не якого-небудь конкретного ресурсу. В цьому випадку стартовий рядок може виглядати так:

OPTIONS \* HTTP / 1.1

Версія визначає, відповідно до якої версією стандарту HTTP складений запит. Вказується як два числа, розділених крапкою (наприклад 1.1).

Для того, щоб звернутися до веб-сторінці за певною адресою (в даному випадку шлях до ресурсу - це «/»), слід відправити наступний запит:

GET / HTTP / 1.1 (1.2)

Host: exzample.google.com

При цьому слід враховувати, що для розриву рядків слід використовувати символ повернення каретки (Carriage Return), за яким слідує символ перекладу рядка (Line Feed). Після оголошення останнього заголовка послідовність символів для розриву рядків додається двічі.

Втім, в специфікації HTTP рекомендується програмувати HTTP-сервер таким чином, щоб при обробці запитів в якості міжрядкового роздільник сприймався символ LF, а попередній символ CR, за наявності такого, ігнорувався. Відповідно, на практиці більшість серверів коректно обробить і такий запит, де заголовки відокремлені символом LF, і він же двічі доданий після оголошення останнього заголовка.

А як отримати відповідь? Стартовий рядок відповіді має наступну структуру:

HTTP / Версія Код стану Пояснення ()

Версія протоколу тут задається так само, як в запиті.

Код стану (Status Code) - три цифри (перша з яких вказує на клас стану), які визначають результат здійснення запиту. Наприклад, в разі, якщо був використаний метод GET, і сервер надає ресурс із зазначеним ідентифікатором, то такий стан задається за допомогою коду 200. Якщо сервер повідомляє про те, що такого ресурсу не існує - 404. Якщо сервер повідомляє про те, що не може надати доступ до цього ресурсу через відсутність необхідних привілеїв у клієнта, то використовується код 403. Специфікація HTTP 1.1 визначає 40 різних кодів HTTP, а також допускається розширення протоколу і використання додаткових кодів станів.

Пояснення до коду стану (Reason Phrase) - текстове (але не включає символи CR і LF) пояснення до коду відповіді, призначене для спрощення читання відповіді людиною. Пояснення може не враховуватися клієнтським програмним забезпеченням, а також може відрізнятися від стандартного в деяких реалізаціях серверного програмного забезпечення.

Після стартового рядка слідують заголовки, а також тіло відповіді. наприклад:

HTTP / 1.1 200 OK

Server: nginx / 1.2.1

Date: Sat, 08 Mar 2014 22:53:46 GMT

Content-Type: application / octet-stream

Content-Length: 7

Last-Modified: Sat, 08 Mar 2014 22:53:30 GMT

Connection: keep-alive

Accept-Ranges: bytes

Wisdom

Тіло відповіді слідує через два розрива рядків після останнього заголовка. Для визначення закінчення тіла відповіді використовується значення заголовка Content-Length (в даному випадку відповідь містить 7 вісімкових байтів: слово «Wisdom» і символ розриву рядків). [https://habr.com/post/215117/#uri]