С целью повышения уровня подготовки программистов в области электроники, ознакомления с принципами построения и функционирования процессоров на аппаратном уровне, а также для формирования представления о классической архитектуре микропроцессоров и языках программирования низкого уровня кафедрой электроники и управляющих систем Харьковского национального университета имени В.Н. Каразина при участии студентов старших курсов была разработана учебная модель цифрового процессора. Она построена на микросхемах малой степени интеграции, позволяет продемонстрировать процесс выполнения программы на уровне отдельных логических элементов и обрабатываемых ими сигналов [1].

Чуть позже на кафедре был создан интерфейсный модуль, который позволил подключать макет процессора к персональному компьютеру для мониторинга его состояния и управления работой в реальном времени. Благодаря разработке этого интерфейса и сопутствующего программного обеспечения появилась возможность:

* загружать в оперативную память более объемные программы, полученные путем трансляции с языка ассемблер;
* отображать на экране монитора текущее состояние модели процессора, содержимое регистров, памяти, состояние шин;
* показывать ход выполнения отдельных команд;
* управлять работой процессора как на уровне выполнения отдельных машинных циклов, так и на уровне выполнения подпрограмм.

Все эти возможности существенно улучшают наглядность макета и позволяют более эффективно использовать его в учебном процессе.

Взаимодействие модели процессора с компьютером через созданный интерфейс осуществляется через параллельный восьмиразрядный LPT-порт [2]. Передача данных из компьютера в макет выполняется по параллельной восьмиразрядной шине, а чтение состояния функциональных узлов модели производится полубайтами с помощью системы мультиплексоров и с использованием служебных регистров LPT-порта. При организации обмена, программа, непосредственно, управляет работой как регистров данных, так и входных и выходных служебных регистров в составе порта. Параллельная передача данных обеспечивает высокое быстродействие системы в целом.

## 2  Проблематика

На протяжении последних лет модель процессора успешно применялась в учебной практике факультета компьютерных наук. Работа с моделью осуществлялась при использовании операционных систем, предоставляющих возможность доступа к аппаратным ресурсам компьютера (из серии операционных систем Windows работа была возможна с версиями Windows 98, Windows Vista, Windows XP).

Но на сегодняшний день все перечисленные выше операционный системы не поддерживаются компанией Майкрософт и постепенно вытесняются из использования. На смену им приходят более новые версии, такие как Windows 7, Windows 8, Windows 10. Во всех этих системах с целью обеспечения безопасности и их устойчивости к вредоносным программам максимально ограничен доступ к аппаратным ресурсам компьютера. Это делает невозможным использование ранее созданной программы по управлению моделью процессора, так как в существующей версии передача данных осуществляется через служебные регистры внешнего порта компьютера. Вероятней всего, последующие операционные системы будут еще больше ограничивать возможности прямого управления портами без использования специально разрабатываемых драйверов, предусматривающих соответствующие правила доступа и ограничения полномочий.

Современная тенденция развития персональных компьютеров определена в первую очередь коммерческими интересами производителей и ориентирована на удовлетворение потребностей наиболее широкого контингента пользователей. Это, в первую очередь, — реализация коммуникативных, мультимедийных и игровых функций. По этой причине в современных компьютерах «исчезли» параллельные порты ввода-вывода. Передача данных в основном осуществляется через USB-порты, а также через порты, в основу работы которых положен радиочастотный принцип передачи информации (технологии Bluetooth и Wi‑Fi). В результате созданное ранее оборудование невозможно подключить к современным ноутбукам и стационарным персональным компьютерам.

В связи с этим возникла необходимость для дальнейшего успешного использования модели процессора разработать новый модуль аппаратного интерфейса и новое программное обеспечение, которое позволило бы управлять аппаратурой при работе с современными компьютерами в операционных системах Windows 7 ... Windows 10.

Макет – заменить на модель цифрового процесора

Макет активно использ в учебном процесс. Макет вызывает огромный інтерес у студент во время проведения лаб работ (2 факльтета, констрктивные решения + наглядность вызивает интерес студ и при активном участии ведется его совершенствование в частности Паша – разработал для данного макета дополнительный модуль оперативной пямяти есмкостью 32 килобайт, кот существенно расширяет возможности самой модели). Констркции интерф регимстров(портов) позволила на прямую подключить к модели процессора жидкокристал индикатор и простейшую клавиатуру преобразовав его таким образом в простейшую модель компа. Благодаря свом возможностям модель позволяет демонстрировать как основыне принциапы работы процессора, так и методы взаимодействия с внешними устройствами, оргонизовывать процессы ввода/вывода данных.

С использованием данной разработки на каф электроники подготовленная цикл лаб работ, которые внедреныв учебный процесс на фак комп наук и физ тех фак ХНУ каразина.

Упомянуть про старую программу, новую программу, но не решена главная задача …

Требует системного изменение программного обеспечения, которое не возможно без разработки нового интерфейсного модуля/

К сожалению в ВУЗАх укрины данная модель является уникальной разработкой и в других вузах нет аналогов.

Метою дипломної роботы э дослідження та аналіз методів організації взаємодії в комп’ютерних системах управління та розробка на результатах цього аналізу нового апаратно-програмного комплексу для моніторинг тауправління моделлю цифрового процесору.

Галузь застосування даного дослідження є учбовий процес з підготовки фахівців по програмуванню та електроніці на технічних факультетах ХНУ каразина.

Основна частина

Опис об’єкту та аналіз можливостей управління

При розробці моделі проц. В її конструкції був передбачений роз’єднувач для підключення комп. Інтерфейсу. На контакти роз’єднав виведено:

* внутрішня шина данних процессора
* шина адреси
* сигнали управління оперативно запам'ятовуючим пристроєм(ОЗП)
* Вихідні шини регістрв АЛУ
* 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]