**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

**Кафедра ЕОМ**



**Звіт з лабораторної роботи №6**

**з дисципліни “** **Комп’ютерні системи ”**

**на тему: ” Побудова функціональної схеми процесорів архітектури RISC CPU.”**

**Виконав: студент .гр. КІ-33**

**Кіндій В. А.**

**Прийняв: викладач**

**Козак Н. Б.**

**Львів 2020 р.**

**Мета роботи:**

Навчитись розробляти нові функціональні модулі з метою розширення функціонування базової архітектури процесорів RISC, що відповідає програмній моделі, а також створення функціональної схеми.

**Завдання:**

Згідно варіанту в журналі та узгоджено з викладачем обрати один з процесорів заданого сімейства на базі RISC архітектури, в звіті навести основну інформацію по процесору, схему та характеристику.

Внести необхідні зміни в проект з 5-ї лабораторної (типи кешів, розмірності шин даних, назви та типи сигналів, операції, буферні модулі і т.д.) з метою реалізації фукнціоналу з обраного процесору. В звіті представити коротко внесені зміни з описом ходу роботи, а також кінцеву функціональну схему програмної моделі.

**Хід роботи:**

Варіант: 7) **Hewlett-Packard's PA-RISC, also known as HP-PA.**

PA-RISC — мікропроцесорна архітектура, розроблена компанією Hewlett-Packard. Як випливає з назви, це RISC-архітектура; PA означає Precision Architecture. Іноді цю архітектуру називають HP/PA (Hewlett Packard Precision Architecture).

Архітектура була представлена ​​26 лютого 1986 року, коли були випущені перші моделі HP 3000 Series 930 та HP 9000 Model 840 оснащені першою реалізацією архітектури - TS1.

HP припинила продажі машин HP 9000, заснованих на PA-RISC, в кінці 2008 року. Підтримка серверів на процесорах PA-RISC продовжена до 2013 року. Передбачається, що нові машини, засновані на Itanium, зможуть замінити PA-RISC.

Особливістю архітектури PA-RISC є внекрістальная реалізація кешу, що дозволяє реалізувати різні обсяги кеш-пам'яті і оптимізувати конструкцію в залежності від умов застосування. Зберігання команд і даних здійснюється в роздільних кешах, причому процесор з'єднується з ними за допомогою високошвидкісних 64-бітових шин. Кеш-пам'ять реалізується на високошвидкісних кристалах статичної пам'яті (SRAM), синхронізація яких здійснюється безпосередньо на тактовій частоті процесора. При тактовій частоті 100 МГц кожен кеш має смугу пропускання 800 Мбайт / с при виконанні операцій зчитування і 400 Мбайт / с при виконанні операцій запису. Мікропроцесор апаратно підтримує різний об'єм кеш-пам'яті: кеш команд може мати обсяг від 4 Кбайт до 1 Мбайт, кеш даних - від 4 Кбайт до 2 Мбайт. Щоб знизити коефіцієнт промахів застосовується механізм хешування адреси. В обох кешах для підвищення надійності застосовуються додаткові контрольні розряди, причому помилки кеша команд коригуються апаратними засобами.

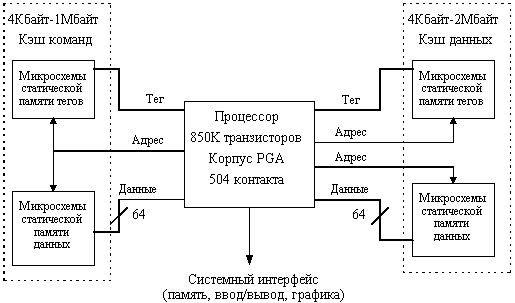


Рис 1. Блок-схема процессора PA 7100

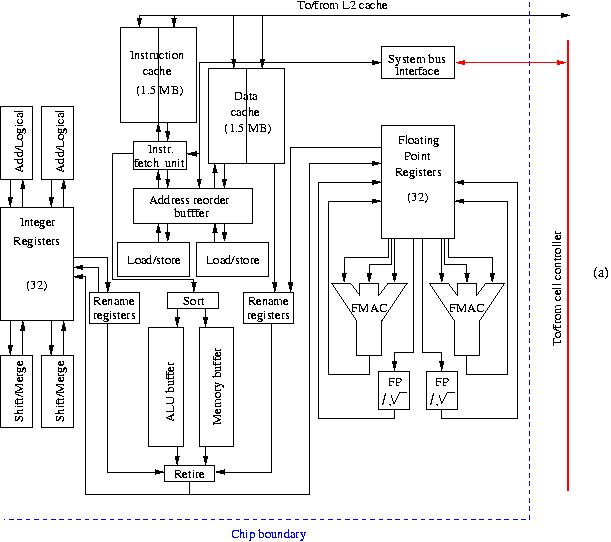


Рис 2. Блок-схема процессора PA 7100

Процесор під'єднується до пам'яті і підсистемі вводу / виводу за допомогою синхронної шини. Процесор може працювати з трьома різними відносинами внутрішньої і зовнішньої тактової частоти в залежності від частоти зовнішньої шини: 1: 1, 3: 2 і 2: 1. Це дозволяє використовувати в системах різні за швидкістю мікросхеми пам'яті.

Конструктивно на кристалі PA-7100 розміщені цілочисельний процесор, процесор для обробки чисел з плаваючою точкою, пристрій управління кешем, уніфікований буфер TLB, пристрій управління, а також ряд інтерфейсних схем. Цілочисельний процесор включає АЛУ, пристрій зсуву, суматор команд переходу, схеми перевірки кодів умов, схеми обходу, універсальний регістровий файл, регістри управління і регістри адресного конвеєра. Пристрій управління кеш-пам'яттю містить регістри, що забезпечують перезавантаження кеш-пам'яті при виникненні промахів і контроль когерентного стану пам'яті. Це пристрій містить також адресні регістри сегментів, буфер перетворення адреси TLB і апаратуру хешування, керуючу перезавантаженням TLB. До складу процесора плаваючою точки входять пристрій множення, арифметико-логічний пристрій, пристрій ділення і витягання квадратного кореня, регістровий файл і схеми "закоротки" результату. Інтерфейсні пристрої включають всі необхідні схеми для зв'язку з кеш-пам'яттю команд і даних, а також з шиною даних. Узагальнений буфер TLB містить 120 рядків асоціативної пам'яті фіксованого розміру і 16 рядків змінного розміру.

Вношу зміни до проекту 5 лабораторної:

1. Розширення кількості операцій в блоці обчислень:

Floating.cpp

|  |
| --- |
| case 7: // AND  opcode\_encode = "FAND";  significant\_result = int(fpua\_significand\_tmp) / int(fpub\_significand\_tmp);  wait();  break;  case 8: // OR  opcode\_encode = "FOR";  significant\_result = int(fpua\_significand\_tmp) / int(fpub\_significand\_tmp);  wait();  break;  case 9: // NOT  opcode\_encode = "FNOT";  significant\_result = ~(int(fpua\_significand\_tmp));  wait();  break; |

1. Додавання прапорців в блоці обчислень

Exec.h

|  |
| --- |
| sc\_out<bool> C; // Carry bit  sc\_out<bool> V; // Overflow bit  sc\_out<bool> Z; // Zero bit  **sc\_out<bool> S; // Sign bit - the value of left bit in register**  **sc\_out<bool> P; // Parity bit - parity control**  sc\_out<signed int> dout; // output data  sc\_out<bool> out\_valid; // output valid  sc\_out<unsigned> destout; // write to which registers?  sc\_in\_clk CLK; |

Main.cpp

|  |
| --- |
| sc\_signal<bool> c("C") ;  sc\_signal<bool> v("V") ;  sc\_signal<bool> z("Z") ;  **sc\_signal<bool> s("S");**  **sc\_signal<bool> p("P");**  sc\_signal<signed> dout("DOUT") ;  sc\_signal<bool> out\_valid("OUTPUT\_VALID") ;  sc\_signal<unsigned> destout("DESTOUT") ; |

3. Зміни в кеші для інструкцій

Icache.h

|  |
| --- |
| **#define BOOT\_LENGTH 6**  **#define MAX\_CODE\_LENGTH 800**  struct icache : sc\_module {  sc\_in<unsigned > datain; // modified instruction  sc\_in<bool> cs; // chip select  sc\_in<bool> we; // write enable for SMC  sc\_in<unsigned > addr; // address  sc\_in<bool> ld\_valid; // load valid  sc\_in<signed> ld\_data; // load data value  sc\_out<unsigned > dataout; // ram data out  sc\_out<bool> icache\_valid; // output valid  sc\_out<bool> stall\_fetch; // stall fetch if busy  sc\_in\_clk CLK; |

4. Зміни в кеші для даних

Dcahce.h

|  |
| --- |
| // initialize instruction dmemory from external file  FILE \*fp = fopen("dcache","r");  int size=0;  int i=0;  unsigned mem\_word;  dmemory = new unsigned[**10000**];  dsmemory = new unsigned[**10000**];  dtagmemory = new unsigned[**10000**];  printf("\*\* ALERT \*\* DCU: initialize Data Cache\n");  while (fscanf(fp,"%x", &mem\_word) != EOF) {  dmemory[size] = mem\_word;  dsmemory[size] = 0;  dtagmemory[size] = size;  size++;  }  for (i=size; i<**10000**; i++) {  dtagmemory[i] = 0xdeadbeef;  dmemory[i] = 0xdeadbeef;  dsmemory[i] = 0;  } |

5. Зміни назви сигналів в DECODE\_BLOCK і EXEC\_BLOCK

Decode.h

|  |
| --- |
| sc\_out<signed int> src\_A; // operand A  sc\_out<signed int> src\_B; // operand B  **sc\_out<bool> EXECUTE\_A; // data forwarding to operand A**  **sc\_out<bool> EXECUTE\_B; // data forwarding to operand B**  sc\_out<bool> stall\_fetch; // stall fetch due to branch  sc\_out<bool> decode\_valid; // decoder output valid |

Exec.h

|  |
| --- |
| sc\_in<signed int> dinb; // operand B  sc\_in<bool> **EXECUTE\_A**; // data forwarding A valid  sc\_in<bool> **EXECUTE\_B**; // data forwarding B valid  sc\_in<unsigned> dest; // destination register number |

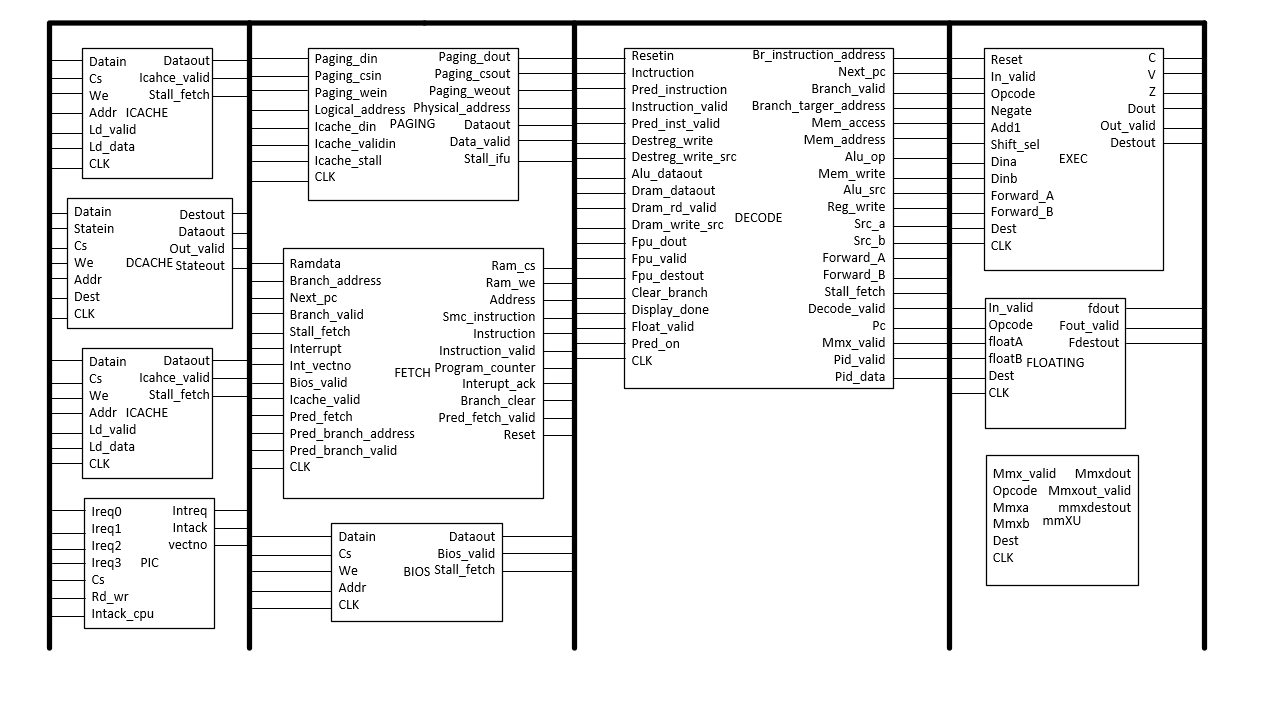


Рис 1. Функціональна схема RISC CPU

Висновок: навчився розробляти нові функціональні модулі з метою розширення функціонування базової архітектури процесорів RISC, що відповідає програмній моделі, а також створення функціональної схеми.