Politechnika Wrocławska

Architektura Systemów Komputerowych

Wykład 6

Dr inż. Radosław Michalski
Katedra Inteligencji Obliczeniowej, Wydział Informatyki i Zarządzania
Politechnika Wrocławska

Wersja 1.1, wiosna 2018

Źródła i licencja

Najbardziej aktualna wersja tego wykładu znajduje się tu:
https://github.com/rmhere/lecture-comp-arch-org

Zawartość tego wykładu

Jak podejrzeć kod asemblera?

Architektura RISC

Architektura MIPS

Jak podejrzeć kod asemblera?

Przykładowy kod źródłowy C++ i asembler (x64)

Załóżmy, że kod źródłowy programu napisano w C++. W jaki sposób można sprawdzić jak będzie wyglądać jego postać w języku asemblera?

hello.cc:

```
#include <iostream>
int main() {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

```
$ g++ -O2 -S hello.cc -o hello.asm
```

Architektura RISC

Wprowadzenie

- architektura CISC
  - instrukcje bardziej złożone
  - zmienna długość instrukcji
  - więcej cykli procesora na wykonanie jednej instrukcji
- architektura RISC
  - instrukcje prostsze i stałej długości
  - szybsze wykonanie elementarnych instrukcji
- obecnie rozróżnienie nie jest już tak oczywiste

Architektura RISC

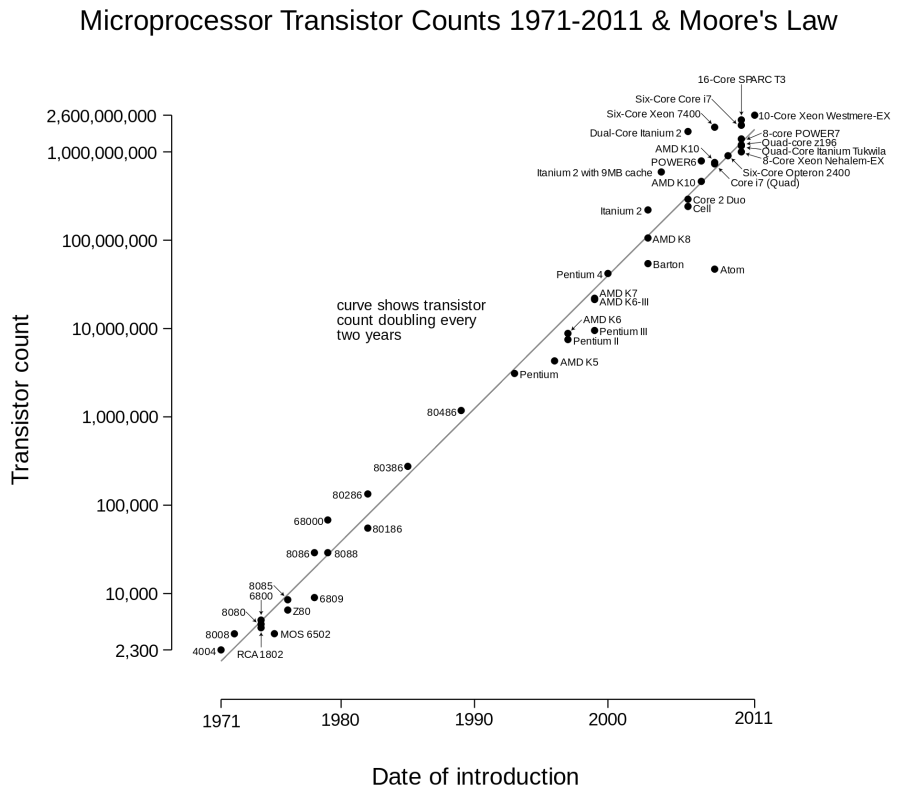
Rys historyczny

- wczesne próby uproszczenia zestawu instrukcji:
  - CDC 6600 (Seymour Cray, 1964)
  - IBM 801 (John Cocke, 1975 - 1980)
- projekt VLSI organizacji DARPA
  - w latach 70-tych procesor składał się ze 100,000 tranzystorów
  - cel: automatyzacja procesu projektowania procesorów
  - efekty projektu: stacje robocze CAD, Stanford University Network, ustandaryzowana specyfikacja Unix (Berkeley Software Distribution), Berkeley RISC, Stanford MIPS

Architektura RISC

Liczba tranzystorów, prawo Moore'a

Liczba tranzystorów - przeczytaj tę stronę Wikipedii



Architektura RISC

Architektury RISC (ISA)

- **Systemy low end i mobilne**
  - ARM®
  - MIPS™
  - Hitachi™ SuperH
  - Atmel® AVR®
  - RISC-V
  - Tensilica Xtensa®
- **Systemy high end**
  - MIPS™
  - IBM® Power®
  - Oracle® SPARC®
  - HP® PA-RISC
  - DEC™ Alpha
  - RISC-V

źródło: https://en.wikipedia.org/wiki/Reduced_instruction_set_computing

Architektura RISC

RISC-V - darmowa i otwarta ISA RISC

- projekt zapoczątkowany w 2010 roku
- powstał na University of California, Berkeley, US
- cel: mała i szybka architektura o niewielkim zużyciu energii
- licencja BSD

Architektura RISC

Popularne ISA

Film

Engineering8 - Top 10 Popular CPU Instruction Set Today

Architektura MIPS

Historia

- **M**icroprocessor without **I**nterlocked **P**ipeline **S**tages - **MIPS™**
- zapoczątkowana na Uniwersytecie Stanforda - John L. Hennessy (1981)
- komercjalizacja przez MIPS Computer Systems (później MIPS) Technologies
- pierwszy układ: R2000 (1985)
- w 2013r. Imagination Technologies przejęło MIPS Technologies
- w 2017r. Imagination Technologies odsprzedało gałąź MIPS firmie Tallwood MIPS Inc.

Architektura MIPS

Założenia architektury

- zestaw prostych instrukcji
- podejście load and store
- proste kodowanie instrukcji
- przetwarzanie potokowe (ang. *pipelining*)

Architektura MIPS

Gdzie ją można znaleźć?

- systemy wbudowane:
  - Nintendo® 64 - NEC VR4300 (bazuje na MIPS R4300i)
  - Sony® PlayStation® 2 - MIPS R5900 “Emotion Engine”
  - Sony® PSP® - MIPS R4000
- superkomputery:
  - SGI® Challenge (R4400, R8000, R10000)

Architektura MIPS

Specyfikacja - MIPS64

- rejestry
  - 32 64-bitowe rejestry ogólnego przeznaczenia
  - 32 64-bitowe rejestry zmiennoprzecinkowe
  - rejestry specjalne, np. rejestr statusu zmiennoprzecinkowego
- typy danych:
  - 8, 16, 32, 64 bity dla liczb całkowitych
  - 32 bity pojedyńczej precyzji dla liczb zmiennoprzecinkowych
  - 64 bity podwójnej precyzji dla liczb zmiennoprzecinkowych
- typy adresowania:
  - immediate (liczba wprost)
  - rejestr
  - base displacement (przesunięcie)
  - pamięć adresowana adresami 64-bitowymi
- **w MIPS32 rejestry mają rozmiar 32 bitów**

Architektura MIPS

Instrukcje

- load and store
  - do każdego rejestru ogólnego przeznaczenia można załadować dane z pamięci (load) lub je z niego pobrać (store)
- jednostka arytmetyczno-logiczna (ALU)
  - instrukcje rejestr-rejestr
- gałęzie i skoki
  - wszystkie odgałęzienia są warunkowe (porównanie instrukcji porównuje dwa rejestry)
- liczby zmiennoprzecinkowe
  - format IEEE 754

Architektura MIPS

Przetwarzanie potokowe (pipeline processing)

- kluczowe rozwiązanie do uczynienia procesorów szybkimi
- wiele instrukcji realizowanych jest jednocześnie
- każdy krok w potoku wykonuje część operacji instrukcji (etap lub segment)
- czas przetwarzania segmentu potoku wynosi na ogół cykl procesora
- zakładając idealne warunki, czas przetwarzania instrukcji w procesorze z przetwarzaniem potokowym wynosi tyle ile na procesorze bez potoków podzielonym przez liczbę segmentów

Architektura MIPS

Przetwarzanie potokowe w RISC

Klasyczny potok RISC:

- pobranie instrukcji - instruction fetch (IF)
- dekodowanie instrukcji - instruction decode/register fetch (ID)
- wykonanie - execute (EX)
- dostęp do pamięci - memory access (MEM)
- zapis do rejestrów - register writeback (WB)

Architektura MIPS

RISC pipeline - schemat

Instr No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Inductiveload, public domain

Architektura MIPS

MIPS64 i MIPS32 - specyfikacje

MIPS64

- Introduction
- Volume I-A: Introduction to the MIPS64 Architecture
- Volume II-A: The MIPS64® Instruction Set Reference Manual

MIPS32

- Introduction
- Introduction to the MIPS32 Architecture
- The MIPS32 Instruction Set
- MIPS32 Instruction Set Quick Reference

Architektura MIPS

Zestaw instrukcji MIPS32

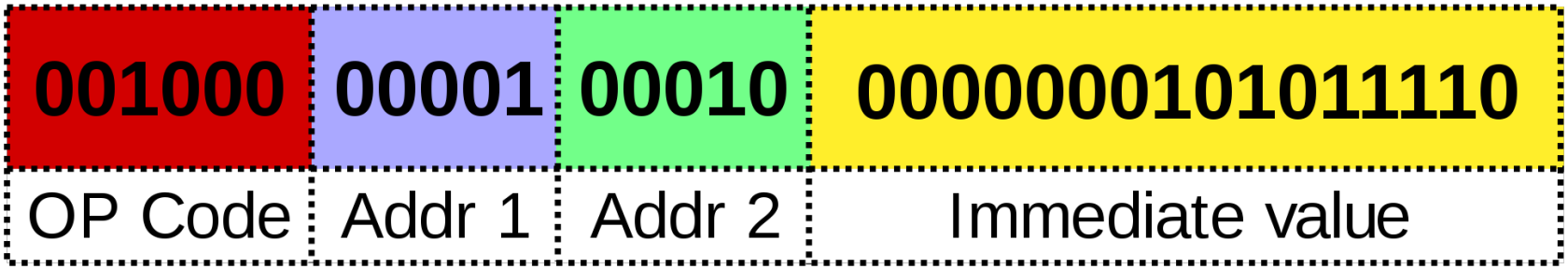
Instrukcje

- długość: stała, 32 bity
- rodzaje: trzy rodzaje instrukcji: R, I, J
- implementacja sprzętowa lub pseudoinstrukcje

Architektura MIPS

Format instrukcji

MIPS32 Add Immediate Instruction



Equivalent mnemonic: **addi $r1, $r2, 350**

Architektura MIPS

Rodzaje instrukcji

- **R-type** - instrukcje dot. rejestrów
- **I-type** - instrukcje dot. wartości podawanych bezpośrednio
- **J-type** - skoki

Architektura MIPS

R-type - operacje na rejestrach

- najbardziej złożony rodzaj
- pracują tylko na rejestrach (wskazujemy ich adresy)

B_{31-26}	B_{25-21}	B_{20-16}	B_{15-11}	B_{10-6}	B_{5-0}
opcode	rejestr s	rejestr t	rejestr d	przesunięcie	funkcja

Tabela 1: Instrukcja R-type

- `add $rd, $rs, $rt`
- `R[d] = R[s] + R[t]`
- 6-bitowy opcode?

Architektura MIPS

I-type - operacje bezpośrednie

B_{31-26}	B_{25-21}	B_{20-16}	B_{15-0}
opcode	rejestr s	rejestr t	immediate

Tabela 2: Instrukcja I-type

- `addi $rd, $rs, immed`
- `R[t] = R[s] + immed`

Architektura MIPS

J-type - jump (skoki)

B_{31-26}	B_{25-0}
opcode	target

Tabela 3: J-type instruction

- `j target`
- `PC <- PC`$_{31\text{-}28}$ `IR`$_{25\text{-}0}$ `00`
- PC (program counter) - rejestr, w którym wskazuje się adres instrukcji do wykonania.
- aktualizacja PC poprzez wykorzystanie czterech jego bitów, następnie 26 bitów podanych w instrukcji uzupełnionych o dwa zera (łącznie 32 bity)

Architektura MIPS

Polecane materiały wideo

- EngMicroLectures - History of ISAs
- MR Trick - How a CPU is made
- Computer History Museum - MIPS: Risking It All on RISC

RISC and MIPS architectures

Źródła i polecane materiały

- Imagination Technologies Limited, MIPS32 Architecture, Hertfordshire, UK (dokumentacja techniczna)
- M. Esponda and R. Rojas, *"The RISC Concept - A Survey of Implementations"*, Freie Universitat Berlin, Berlin, Germany (raport)
- K. Keville, *"Introduction to RISC-V"*, R&D Labs at MIT, 2016 Stanford HPC Conference (wideo)
- J. Kwiatkowski, *"Computer Architecture and Organization"*, Wrocław University of Science and Technology (materiały do kursu)