Politechnika Wrocławska

Architektura Systemów Komputerowych

Wykład 7

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

CPU, FPGA, ASIC

Asembler MIPS

CPU, FPGA, ASIC

Czym układy FPGA różnią się od mikroprocesorów?

- **f**ield-programmable **g**ate **a**rray - **FPGA**
- pomysł narodził się w latach 80. XX wieku
- cel: zbudowanie reprogramowalnego układu
- brak stałych konfiguracji bramek
- bardziej elastyczna architektura
- hardware description language (HDL) do specyfikacji konfiguracji

CPU, FPGA, ASIC

Zastosowania układów FPGA

- niegdyś: prototypowanie, niewielkie serie
- obecnie: niemal wszędzie
  - elektronika użytkowa
  - zastosowania medyczne
  - przetwarzanie wideo, obrazów i dźwięku
  - branża automotive
  - wiele, wiele więcej

CPU, FPGA, ASIC

ASIC

- **a**pplication-**s**pecific **i**ntegrated **c**ircuit - **ASIC**
- tworzony do wykonywania konkretnego zadania
- cel musi być znany przed rozpoczęciem projektowania i produkcji
- stała konfiguracja (bezpieczeństwo)
- zastosowania
  - branża kosmiczna (większa odporność na promieniowanie)
  - DSP
  - kryptografia

CPU, FPGA, ASIC

Porównanie

- **CPU**: najdroższe, wysokie zużycie energii, stała konfiuracja sprzętowa, ogólnego przeznaczenia
- **FPGA**: tańsze, duża liczba bramek, logika do zaprogramowania, bardziej elastyczne, także używane do prototypowania układów ASIC
- **ASIC**: najtańsze (w produkcji), konkretne zastosowanie, stała konfiguracja sprzętowa, niskie zużycie energii

CPU, FPGA, ASIC

Porównanie - wydobywanie Bitcoinów

Satoshi, *public domain*

bitcoinwiki - Non-specialized hardware comparison
bitcoinwiki - Mining hardware comparison

Asembler MIPS

Kompilacja, asemblacja, interpretacja

- kompilacja - plik wykonywalny (kod maszynowy)
- asemblacja - plik wykonywalny (kod maszynowy)
- interpretacja - linia po linii (interpreter)

Asembler MIPS

Struktura programu

- plik tekstowy
- na początku deklaracja danych
- następnie kod programu

Asembler MIPS

Struktura programu c.d.

Deklaracja danych:

- poprzedzona dyrektywą asemblera `.data`
- deklaracja danych używanych w programie
- umieszczone one będą w głównej pamięci (RAM)

Instrukcje:

- poprzedzone dyrektywą asemblera `.text`
- tradycyjnie pierwsza instrukcja po etykiecie `main:`

Komentarze:

- `# rozpoczyna komentarz`

Asembler MIPS

Struktura programu - przykład

```
# Komentarz informujący o nazwie programu,
# twórcy, przeznaczeniu, wersji itp.
# Template.s
# Struktura programu asemblera MIPS

    .data           # deklaracje danych po tym wierszu
                    #
    .text           # instrukcje po tym wierszu
    main:           # etykieta - początek bloku instrukcji

# Koniec programu, pozostaw pusty wiersz na końcu.
```

Asembler MIPS

Instrukcje asemblera MIPS

- każdy wiersz zawiera jedną instrukcję
- stały porządek i liczba argumentów
- jeśli potrzeba więcej argumentów w języku wysokego poziomu
  - więcej instrukcji w języku asemblera

Asembler MIPS

Zmienne?

- brak zmiennych, praca na rejestrach (load and store)
- deklaracje danych odnoszą się do pamięci

Asembler MIPS

Bajty i słowa

- bajt - 8 bitów
- słowo to podstawowa porcja informacji, na której operuje system komputerowy
- word (słowo) - 4 bajty - 32 bity (MIPS)

Asembler MIPS

Deklaracja danych

Format:

```
etykieta:    typ    wartość(i)
```

Przykład:

```
myint:   .word    3          # liczba całkowita
array1:  .byte    'a','b'    # dwuelementowa macierz char
array2:  .space   40         # 40 bajtów ciągiem
```

Asembler MIPS

Rejestry

Numer	Nazwa	Opis
$0	$zero	Zawsze zero
$1	$at	Zarezerwowany dla asemblera
$2-$3	$v0, $v1	pierwsza i druga wartość powrotu
$4-$7	$a0...$a3	Pierwsze cztery argumenty funkcji
$8-$15	$t0...$t7	Rejestry tymczasowe
$16-$23	$s0...$s7	Rejestry zapisane
$24-$25	$t8, $t9	Kolejne rejestry tymczasowe
$26-$27	$k0, $k1	Zarezerwowane dla systemu operacyjnego
$28	$gp	Wskaźnik globalny
$29	$sp	Wskaźnik stosu
$30	$fp	Wskaźnik ramki
$31	$ra	Adres powrotu

Asembler MIPS

Wywołania systemowe

- używane do odczytu lub wyświetlenia wartości lub ciągów z I/O i informacji o zakończeniu wykonania
- syscall do wywołania
- najpierw ustawienie odpowiednich wartości w rejestrach $v0, $a0-$a1, $f
- wartość wyniku działania syscall (jeśli jakakolwiek) w $v0

Asembler MIPS

Wywołania systemowe - lista

Usługa	Wartość w $v0	Argumenty	Wynik
print_int	1	$a0 = liczba całkowita do wyświetlenia	
print_float	2	$f12 = float do wyświetlenia	
print_double	3	$f12 = double do wyświetlenia	
print_string	4	$a0 = adres ciągu w pamięci	
read_int	5		l. całkowita pobrana do $v0
read_float	6		float pobrany do $v0
read_double	7		double pobrany do $v0
read_string	8	$a0 = ades pamięci bufora na ciąg znaków, $a1 = długość bufora na ciąg (n)	
sbrk	9	$a0 = liczba	adres w $v0
exit	10		

Asembler MIPS

Przykład kodu w języku asemblera

Napisz program wykonujący następującą operację:

`(a + b) - (c - d)`

Asembler MIPS

Przykład kodu w języku asemblera

```
.data
    a: .word 5
    b: .word 6
    c: .word 4
    d: .word 3

.text
main:
    lw $t0, a
    lw $t1, b
    lw $t2, c
    lw $t3, d

    add $t4, $t0, $t1
    sub $t5, $t2, $t3
    sub $t6, $t4, $t5

    li $v0, 1
    add $a0, $zero, $t6
    syscall
```

Asembler MIPS

MARS - demo

Demonstracja pracy w Symulatorze MIPS - MARS.

Asembler MIPS

Źródła i materiały dodatkowe

- D. Reed, MIPS Architecture and Assembly Language Overview, University of Illinois at Chicago, IL, United States (materiały do kursu)
- Y. Kreinin, How FPGAs work, and why you'll buy one (artykuł)
- R. Baruch, Bulding a CPU on an FPGA (materiały wideo)