Architektura systemów komputerowych

Mariusz Wiśniewski

Politechnika Świętokrzyska w Kielcach
Katedra Informatyki

Lista instrukcji procesora

Plan wykładu

1. Rozkaz, lista rozkazów procesora.
2. Mikroprogramowanie.
3. Język maszynowy.
4. Projekt μP: koncepcja, model rozkazu.

Cele

Architektura procesorów: von Neumana i harwardzka. Wiedza na temat interakcji mikroprocesora, pamięci i urządzeń we/wy. Znajomość technik projektowania procesora.

Rozkaz, lista rozkazów procesora

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Architektura systemu komputerowego często jest zdeterminowana modelem rozpoznawanych poleceń – w takim przypadku mówi się o specyfikacji ISA (ang. Instruction Set Architecture).

Definicje:
- rozkaz
- modele ISA
- typy rozkazów

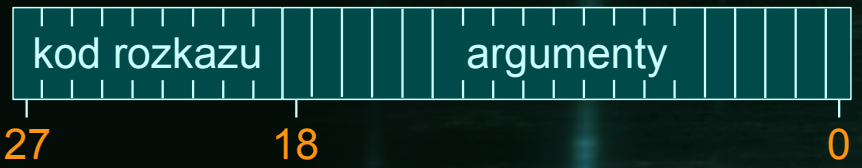
Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

Architektura systemu komputerowego często jest zdeterminowana modelem rozpoznawanych poleceń – w takim przypadku mówi się o specyfikacji ISA (ang. Instruction Set Architecture).

Definicje:
- rozkaz
- modele ISA
- typy rozkazów

W odniesieniu do CPU rozkaz jest kodem operacji, wykonywanej przez procesor. Postać rozkazu jest następująca:



Zarówno pole kodu rozkazu, jak i pole argumentów są umowne – w zależności od typu rozkazu poszczególne bity mogą mieć różne znaczenia. Z tego powodu wyróżnia się dwa rodzaje rozkazów:
- bezadresowe – w których operandy są domyślne,
- adresowe – dla których należy podać adresy argumentów.

Jedynym stałym polem rozkazu jest jego kod. Pozostałe elementy mogą być stałe lub zmienne. Z tego względu rozkazy można podzielić na:
- instrukcje o stałej długości – domena RISC,
- instrukcje o zmiennej długości – często związane z modelem CISC.

M.Wiśniewski

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Architektura systemu komputerowego często jest zdeterminowana modelem rozpoznawanych poleceń – w takim przypadku mówi się o specyfikacji ISA (ang. Instruction Set Architecture).

Definicje:
- rozkaz
- modele ISA
- typy rozkazów

W odniesieniu do CPU rozkaz jest kodem operacji, wykonywanej przez procesor. Postać rozkazu jest następująca:

Zarówno pole kodu rozkazu, jak i pole argumentów są umowne – w zależności od typu rozkazu poszczególne bity mogą mieć różne znaczenia. Z tego powodu wyróżnia się dwa rodzaje rozkazów:
- bezadresowe – w których operandy są domyślne,
- adresowe – dla których należy podać adresy argumentów.

Jedynym stałym polem rozkazu jest jego kod. Pozostałe elementy mogą być stałe lub zmienne. Z tego względu rozkazy można podzielić na:
- instrukcje o stałej długości – domena RISC,
- instrukcje o zmiennej długości – często związane z modelem CISC.

Wyróżnia się dwa modele ISA:
- CISC – ang. Complex Instruction Set Computer:
  - duża liczba różnorodnych rozkazów,
  - wykonanie instrukcji może zająć wiele cykli zegara.
- RISC – ang. Reduced Instruction Set Computer:
  - mała liczba rozkazów (typowo do 32),
  - zazwyczaj każda instrukcja wykonuje się w jednym cyklu zegara,
  - łatwa implementacja potoku.

M.Wiśniewski

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Architektura systemu komputerowego często jest zdeterminowana modelem rozpoznawanych poleceń – w takim przypadku mówi się o specyfikacji ISA (ang. Instruction Set Architecture).

Definicje:
- rozkaz
- modele ISA
- typy rozkazów

W odniesieniu do CPU rozkaz jest kodem operacji, wykonywanej przez procesor. Postać rozkazu jest następująca:

Zarówno pole kodu rozkazu, jak i pole argumentów są umowne – w zależności od typu rozkazu poszczególne bity mogą mieć różne znaczenia. Z tego powodu wyróżnia się dwa rodzaje rozkazów:
- bezadresowe – w których operandy są domyślne,
- adresowe – dla których należy podać adresy argumentów.

Jedynym stałym polem rozkazu jest jego kod. Pozostałe elementy mogą być stałe lub zmienne. Z tego względu rozkazy można podzielić na:
- instrukcje o stałej długości – domena RISC,
- instrukcje o zmiennej długości – często związane z modelem CISC.

Instrukcje można podzielić na grupy:
- rozkazy przetwarzania danych i operacji na pamięci,
- operacje arytmetyczne i logiczne,
- instrukcje modyfikujące przepływ sterowania.

Powyższe grupy rozkazów są związane z konkretnymi blokami mikroarchitektury CPU.

Wyróżnia się dwa modele ISA:
- CISC – ang. Complex Instruction Set Computer:
  - duża liczba różnorodnych rozkazów,
  - wykonanie instrukcji może zająć wiele cykli zegara.
- RISC – ang. Reduced Instruction Set Computer:
  - mała liczba rozkazów (typowo do 32),
  - zazwyczaj każda instrukcja wykonuje się w jednym cyklu zegara,
  - łatwa implementacja potoku.

M.Wiśniewski

Rozkaz, lista rozkazów

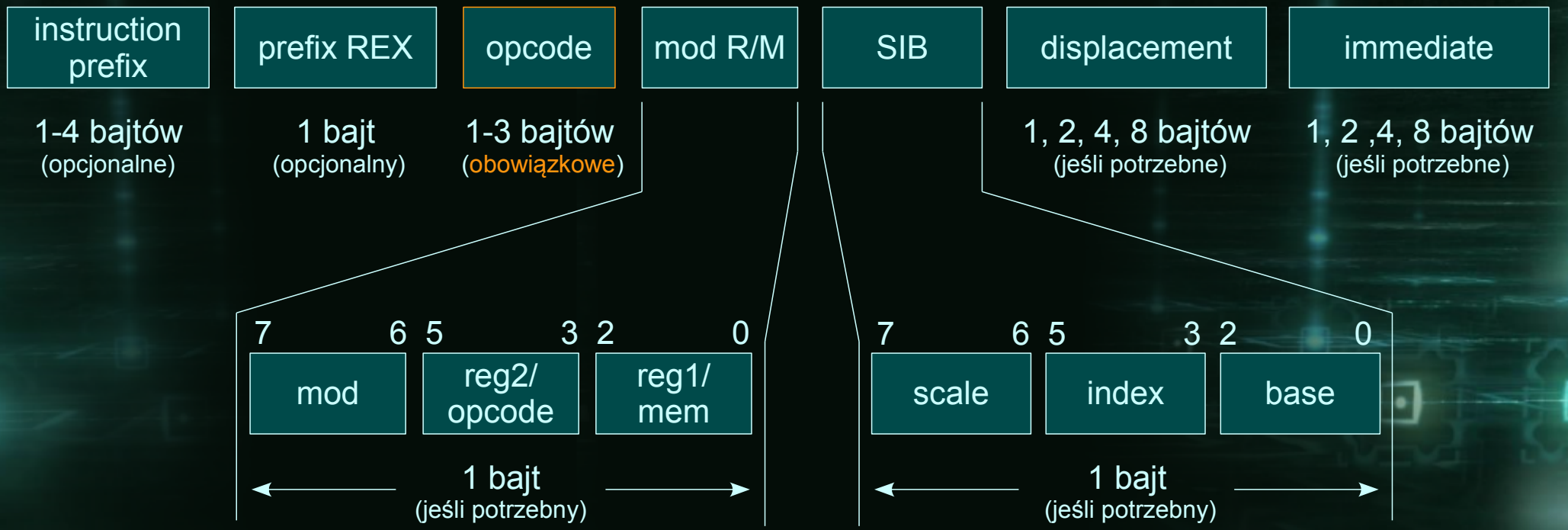
pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Forma rozkazów jest charakterystyczna dla rodziny procesora i jest związana z modelem funkcjonalnym architektury.

Rozkazy IA-32 (x86) mogą składać się od 1 do 17 bajtów. Rozkaz może zawierać wartość adresu w pamięci lub liczbę, jaką procesor zapisze do rejestru.

Format rozkazu:



M.Wiśniewski

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Forma rozkazów jest charakterystyczna dla rodziny procesora i jest związana z modelem funkcjonalnym architektury.

Rozkazy IA-32 (x86) mogą składać się od 1 do 17 bajtów. Rozkaz może zawierać wartość adresu w pamięci lub liczbę, jaką procesor zapisze do rejestru.

Znaczenie pól rozkazu:

instruction prefix

Instrukcja może mieć jeden prefiks z każdej z grup:
- grupa 1: lock, repe/repz, rep, repne/repnz
- grupa 2: CS, DS, ES, FS, GS, SS, znacznik skoku
- grupa 3: zmiana rozmiaru operandów
- grupa 4: zmiana rozmiaru adresu

prefix REX

Dostępny tylko w trybie 64-bitowym. Włącza dostęp do dodatkowych rejestrów: RAX, RBX, RIP, RSP, RFLAGS, …, ośmiu rejestrów całkowitoliczbowych R8...R15, ośmiu dodatkowych rejestrów SSE, kontrolnych CR8 do CR15 i debuggera DR8 do DR15.

mod R/M

Bity określają rodzaj operandów, gdzie reg1 jest źródłem, a reg2 miejscem przeznaczenia. Operandami może być:
- jeden rejestr,
- dwa rejestry,
- rejestr – adres lub pamięć – adres,
- kombinacja rejestrów, np. BX + SI, BP + DI,
- rejestr + [rejestr] + przemieszczenie – 0, 1, 2 bajty.

SIB

Jest prefiksem sposobu adresowania, w którym do obliczenia adresu procesor stosuje wzór:

$$(\text{rejestr indeksowy} * 2^{\text{skala}}) + \text{rejestr bazowy}$$

Mechanizm adresowania uwzględnia pole mod R/M, dzięki czemu adres może mieć postać w której dostępne jest również przemieszczenie, np.:

[EBX * 8 + ESI + 200h]

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Format rozkazu zależy od wielu czynników. Należy wziąć pod uwagę: rodzaje wykonywanych operacji, długość słowa maszynowego, dostępne moduły, itp.

Definicje:
- model programowania
- implementacja
- liczba operandów

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Format rozkazu zależy od wielu czynników. Należy wziąć pod uwagę: rodzaje wykonywanych operacji, długość słowa maszynowego, dostępne moduły, itp.

Definicje:
- model programowania
- implementacja
- liczba operandów

Format instrukcji jest zależny od środowiska, dla którego jest przeznaczony CPU. Należy tu wziąć pod uwagę:
- rodzaj danych,
- model komunikacji między CPU z pamięcią,
- zasoby wbudowane w CPU,
- dostępne zasoby komputera.

M.Wiśniewski

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

- wstęp
- model IA-32
- zbiór rozkazów

Format rozkazu zależy od wielu czynników. Należy wziąć pod uwagę: rodzaje wykonywanych operacji, długość słowa maszynowego, dostępne moduły, itp.

Definicje:

- model programowania
- implementacja
- liczba operandów

Format instrukcji jest zależny od środowiska, dla którego jest przeznaczony CPU. Należy tu wziąć pod uwagę:

- rodzaj danych,
- model komunikacji między CPU z pamięcią,
- zasoby wbudowane w CPU,
- dostępne zasoby komputera.

Implementacja modelu rozkazów może zostać zrealizowana w dwojaki sposób:

- poprzez wykonanie układów kontrolera i dekodera – układy FSM – dedykowanych dla konkretnych rozkazów i dostosowanych do wykonywania ich w całości,
- projektując mikroarchitekturę – wtedy CPU składa się z bloków (zazwyczaj projektowanych osobno), a rozkaz jest dzielony na elementarne operacje, często wspólne dla wielu rozkazów.

Mikroarchitekturę CPU projektuje się stosując model RTL (ang. Register Transfer Level).

M.Wiśniewski

Rozkaz, lista rozkazów

pojęcie rozkazu i listy instrukcji procesora

– wstęp

– model IA-32

– zbiór rozkazów

Format rozkazu zależy od wielu czynników. Należy wziąć pod uwagę: rodzaje wykonywanych operacji, długość słowa maszynowego, dostępne moduły, itp.

Definicje:
- model programowania
- implementacja
- liczba operandów

Zazwyczaj rozkazy realizują operacje obliczeniowe lub kontrolne, związane z operandami (parametrami). Dla architektury typu SISD wyróżnia się następujące typy rozkazów:
- 0-operandów – rozkazy bezadresowe – np. operacje wykonują obliczenia wykorzystując stos (koprocesor arytmetyczny),
- 1-operand – np. rozkazy adresowane domyślnie – typowo drugim argumentem (jeśli taki występuje) jest akumulator,
- 2-operandy – model stosowany powszechnie w CPU typu CISC i RISC – zazwyczaj jeden z operandów rozkazu jest jednocześnie argumentem i miejscem na wynik,
- 3-operandy – w zasadzie stosowane tylko w CISC.

W architekturach SIMD liczba argumentów rozkazów zazwyczaj przekracza 3 – tego typu instrukcje realizują przetwarzanie danych wektorowych.

W modelu VLIW (będący implementacją architektury MIMD) rozkazy mogą mieć cechy architektur SISD i SIMD.

Format instrukcji jest zależny od środowiska, dla którego jest przeznaczony CPU. Należy tu wziąć pod uwagę:
- rodzaj danych,
- model komunikacji między CPU z pamięcią,
- zasoby wbudowane w CPU,
- dostępne zasoby komputera.

Implementacja modelu rozkazów może zostać zrealizowana w dwojaki sposób:
- poprzez wykonanie układów kontrolera i dekodera – układy FSM – dedykowanych dla konkretnych rozkazów i dostosowanych do wykonywania ich w całości,
- projektując mikroarchitekturę – wtedy CPU składa się z bloków (zazwyczaj projektowanych osobno), a rozkaz jest dzielony na elementarne operacje, często wspólne dla wielu rozkazów.

Mikroarchitekturę CPU projektuje się stosując model RTL (ang. Register Transfer Level).

Mikroprogramowanie

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

W ogólności sterowanie w układach cyfrowych polega na wygenerowaniu sygnałów sterujących, będących funkcją stanu układu. Owa funkcja jest postaci:

$$C = F(S, t)$$

gdzie:

$C = \{c_0, c_1, \ldots c_n\}$ – jest wektorem sterującym (funkcją sterującą)

$S = \{s_0, s_1, \ldots s_m\}$ – jest wektorem stanu maszyny,

t – oznacza czas (sygnał zegara).

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

W ogólności sterowanie w układach cyfrowych polega na wygenerowaniu sygnałów sterujących, będących funkcją stanu układu. Owa funkcja jest postaci:

$$C = F(S, t)$$

gdzie:

$C = \{c_0, c_1, \ldots c_n\}$ – jest wektorem sterującym (funkcją sterującą)

$S = \{s_0, s_1, \ldots s_m\}$ – jest wektorem stanu maszyny,

t – oznacza czas (sygnał zegara).

Definicje:

- mikrooperacja
- mikroprogram

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

W ogólności sterowanie w układach cyfrowych polega na wygenerowaniu sygnałów sterujących, będących funkcją stanu układu. Owa funkcja jest postaci:

$$C = F(S, t)$$

gdzie:

$C = \{c_0, c_1, \ldots c_n\}$ – jest wektorem sterującym (funkcją sterującą)

$S = \{s_0, s_1, \ldots s_m\}$ – jest wektorem stanu maszyny,

t – oznacza czas (sygnał zegara).

Definicje:

- mikrooperacja
- mikroprogram

Uporządkowany zbiór sygnałów sterujących **C**, w czasie **t**, nazywa się mikrooperacją. Maksymalna liczba mikrooperacji w systemie wynosi 2^n, gdzie n jest liczbą sygnałów sterujących.

Na etapie projektowania układu cyfrowego sygnały sterujące można przypisać do:

- modułów układu,
- wejść układu.

Zazwyczaj interakcje między modułami maszyny cyfrowej opisuje się za pomocą grafu skierowanego, którego węzły odpowiadają modułom. Krawędzie w grafie oznaczają możliwość przesyłu danych między modułami, które łączą. Wagi na krawędziach odpowiada sygnałom sterującym.

M.Wiśniewski

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

W ogólności sterowanie w układach cyfrowych polega na wygenerowaniu sygnałów sterujących, będących funkcją stanu układu. Owa funkcja jest postaci:

$$C = F(S, t)$$

gdzie:

$C = \{c_0, c_1, \ldots c_n\}$ – jest wektorem sterującym (funkcją sterującą)

$S = \{s_0, s_1, \ldots s_m\}$ – jest wektorem stanu maszyny,

t – oznacza czas (sygnał zegara).

Definicje:

- mikrooperacja
- mikroprogram

Uporządkowany zbiór sygnałów sterujących **C**, w czasie **t**, nazywa się mikrooperacją. Maksymalna liczba mikrooperacji w systemie wynosi 2^n, gdzie n jest liczbą sygnałów sterujących.

Na etapie projektowania układu cyfrowego sygnały sterujące można przypisać do:

- modułów układu,
- wejść układu.

Zazwyczaj interakcje między modułami maszyny cyfrowej opisuje się za pomocą grafu skierowanego, którego węzły odpowiadają modułom. Krawędzie w grafie oznaczają możliwość przesyłu danych między modułami, które łączą. Wagi na krawędziach odpowiada sygnałom sterującym.

Jest to uporządkowany ciąg mikrooperacji, których wykonanie powoduje także wykonanie rozkazu. Mikroprogram powstaje na podstawie diagramów instrukcji, tworzonych przez uporządkowanie względem czasu sygnałów sterujących odpowiadających rozkazom.

Mikroprogram musi uwzględniać fazy cyklu rozkazu. Zależnie od modelu mikrooperacji w mikroprogramie mogą być dostępne rozgałęzienia, tak, jak w zwykłym programie – co pozwala wykonywać złożone operacje.

W CPU mikroprogram znajduje się pamięci wbudowanej w układ procesora.

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Mikroprogramowanie

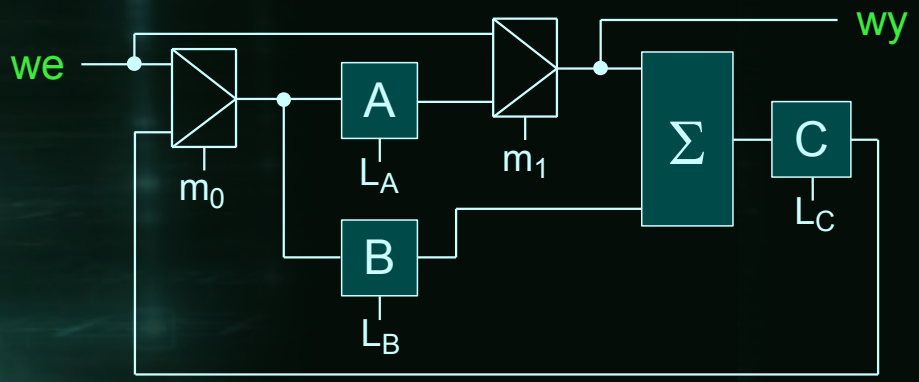
zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Diagram rozkazu wyznacza się na podstawie układu, który ma wykonać instrukcję.



rozkaz	m_0	m_1	L_A	L_B	L_C
A = A + B	-	1	-	-	-
	-	1	-	-	⌟‾
	1	-	⌟‾	-	-
B = we	0	-	-	⌟‾	-

Mikroprogramowanie

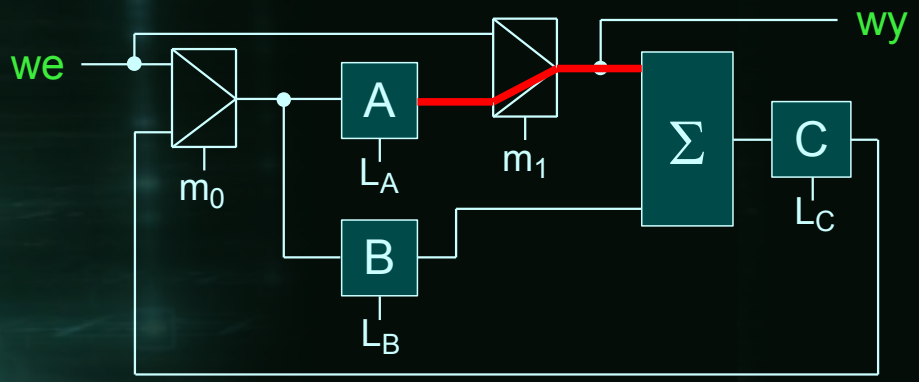
zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Diagram rozkazu wyznacza się na podstawie układu, który ma wykonać instrukcję.



rozkaz	m_0	m_1	L_A	L_B	L_C
A = A + B	-	1	-	-	-
	-	1	-	-	⌐
	1	-	⌐	-	-
B = we	0	-	-	⌐	-

Mikroprogramowanie

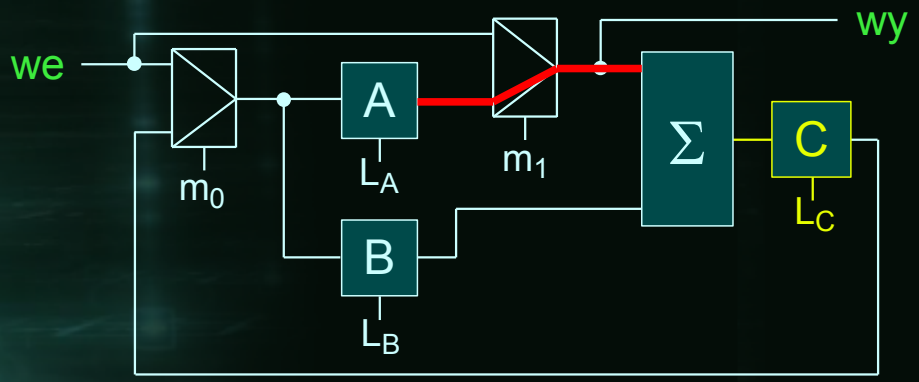
zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Diagram rozkazu wyznacza się na podstawie układu, który ma wykonać instrukcję.



rozkaz	m_0	m_1	L_A	L_B	L_C
A = A + B	-	1	-	-	-
	-	1	-	-	⌐
	1	-	⌐	-	-
B = we	0	-	-	⌐	-

Mikroprogramowanie

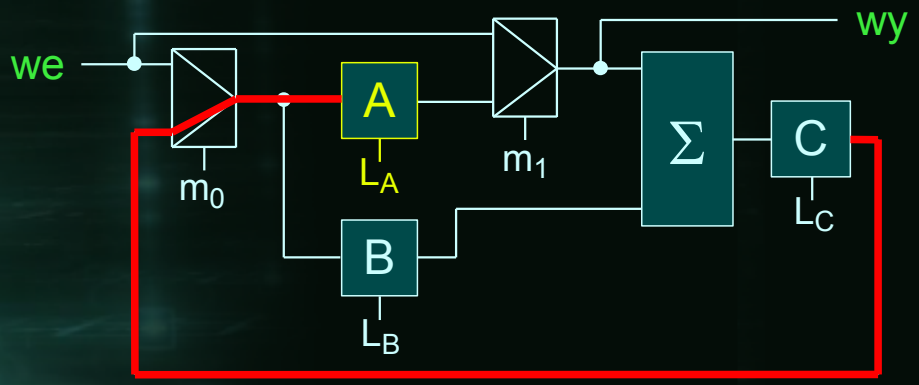
zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Diagram rozkazu wyznacza się na podstawie układu, który ma wykonać instrukcję.



rozkaz	m_0	m_1	L_A	L_B	L_C
A = A + B	-	1	-	-	-
	-	1	-	-	⌜
	1	-	⌜	-	-
B = we	0	-	-	⌜	-

Mikroprogramowanie

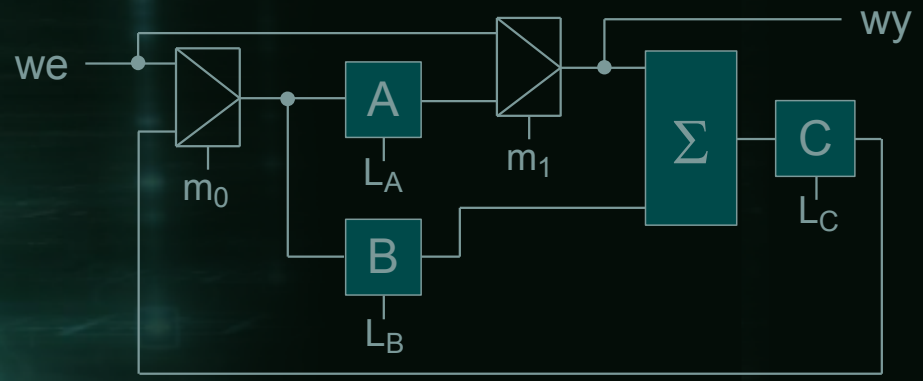
zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Mikrooperacje mogą mieć mniej lub bardziej złożoną postać. Minimalna postać wymaga obecności kodu operacji oraz pola sygnałów sterujących, Bardziej złożone formy pozwalają na tworzenie złożonych programów.

Format mikrooperacji:

Diagram rozkazu wyznacza się na podstawie układu, który ma wykonać instrukcję.



rozkaz	m_0	m_1	L_A	L_B	L_C
A = A + B	-	1	-	-	-
	-	1	-	-	⎍
	1	-	⎍	-	-
B = we	0	-	-	⎍	-

Pole modyfikacji adresu – umożliwia realizację rozgałęzień w mikrokodzie, pozwalając na realizację rozkazów z wariantami.

Pole zawiera adres następnej mikrooperacji, jaką należy wykonać po bieżącej.

Pole zawiera bity kontrolne – zazwyczaj wykonuje się sprawdzanie parzystości.

Pole określa jak dużo cykli zegara zajmie wykonanie danej mikrooperacji.

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Każdy rodzaj CPU ma swoje zalety i wady. Zazwyczaj docelowy model architektury jest uzależniany od przeznaczenia procesora.

Porównanie:
- zalety RISC
- zalety CISC

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

- wstęp
- mikrooperacje
- CISC i RISC

Każdy rodzaj CPU ma swoje zalety i wady. Zazwyczaj docelowy model architektury jest uzależniany od przeznaczenia procesora.

Porównanie:
- zalety RISC
- zalety CISC

Zalety RISC w stosunku do mikrokodu CISC:
- ze względu na model programowania wysokopoziomowego, stosowanie złożonych instrukcji jest mniej opłacalne – zwiększa się rola kompilatorów,
- wydajność CPU może być większa w przypadku prostszych instrukcji – udział % złożonych instrukcji w kodzie wynikowym może być niewielki,
- proste instrukcje można wykonywać bezpośrednio przez dedykowane komponenty,
- jeśli udział złożonych instrukcji w kodzie programu jest niewielki, to zasoby sprzętowe CPU nie są w pełni wykorzystywane,
- złożone instrukcje mogą wymagać wielu cykli zegara oraz zazwyczaj ta liczba nie jest stała, co utrudnia realizację przetwarzania potokowego.

Obecnie procesory z architekturą opartą na mikrokodzie posiadają wbudowane różne układy zwiększające wydajność. Jednocześnie pewna liczba rozkazów jest implementowana zgodnie ze specyfikacją RISC – często procesor potrafi wykonać wiele prostych instrukcji w jednym cyklu zegara.

M.Wiśniewski

Mikroprogramowanie

zagadnienia związane z mikroprogramowaniem

– wstęp

– mikrooperacje

– CISC i RISC

Każdy rodzaj CPU ma swoje zalety i wady. Zazwyczaj docelowy model architektury jest uzależniany od przeznaczenia procesora.

Porównanie:
- zalety RISC
- zalety CISC

Zalety CISC w stosunku do architektury RISC:
- w CPU opartym w całości na mikrokodzie moduły procesora mogą wykonywać czynności, do których nie są dedykowane – np. ALU może służyć również do obliczania adresu efektywnego,
- kompilatory chętnie wykorzystują instrukcje, których operandy odwołują się do pamięci – takie rozkazy ułatwiają generowanie kodu wynikowego, ponadto nie utrudniają zbytnio organizacji potoku,
- złożone rozkazy wykonują więcej operacji/obliczeń, co zmniejsza długość kodu wynikowego,
- nowoczesne procesory CISC/RISC – np. konstrukcje x86 – dekodują instrukcje korzystając z buforowania mikro-operacji, dzięki czemu można jednocześnie wykonywać różne, niewykluczające się czynności.
- proste instrukcje CISC procesory również wykonują w trakcie jednego cyklu zegara.

Zalety RISC w stosunku do mikrokodu CISC:
- ze względu na model programowania wysokopoziomowego, stosowanie złożonych instrukcji jest mniej opłacalne – zwiększa się rola kompilatorów,
- wydajność CPU może być większa w przypadku prostszych instrukcji – udział % złożonych instrukcji w kodzie wynikowym może być niewielki,
- proste instrukcje można wykonywać bezpośrednio przez dedykowane komponenty,
- jeśli udział złożonych instrukcji w kodzie programu jest niewielki, to zasoby sprzętowe CPU nie są w pełni wykorzystywane,
- złożone instrukcje mogą wymagać wielu cykli zegara oraz zazwyczaj ta liczba nie jest stała, co utrudnia realizację przetwarzania potokowego.

Obecnie procesory z architekturą opartą na mikrokodzie posiadają wbudowane różne układy zwiększające wydajność. Jednocześnie pewna liczba rozkazów jest implementowana zgodnie ze specyfikacją RISC – często procesor potrafi wykonać wiele prostych instrukcji w jednym cyklu zegara.

Język maszynowy

Język maszynowy

rodzaje rozkazów a architektura procesora

– asembler

– przykłady

Program dla procesora można wygenerować w wyniku kompilacji języka wysokiego poziomu lub przez zapis kodu maszynowego

Program:
- język asemblera
- proces kompilacji

M.Wiśniewski

Język maszynowy

rodzaje rozkazów a architektura procesora

– asembler

– przykłady

Program dla procesora można wygenerować w wyniku kompilacji języka wysokiego poziomu lub przez zapis kodu maszynowego

Program:
- język asemblera
- proces kompilacji

Kod maszynowy składa się liczb, będących fragmentami rozkazów procesora. W celu ułatwienia programowania instrukcje i ich operandy mają przypisane nazwy symboliczne.

instrukcja	działanie	operandy
mov a, b	a = b	reg, mem, nat
add a, b	a = a + b	reg, mem, nat
adc a, b	a = a + b*	reg, mem, nat
sub a, b	a = a – b	reg, mem, nat
sbb a, b	a = a – b*	reg, mem, nat
and a, b	a = a & b	reg, mem, nat
or a, b	a = a | b	reg, mem, nat
xor a, b	a = a ^ b	reg, mem, nat
not a	a = ~a	reg, mem
call x	przekazanie sterowania pod x i umieszczenie adresu powrotu na stosie	
jmp x	przekazanie sterowania pod x	

* - z przeniesieniem/pożyczką.

M.Wiśniewski

Język maszynowy

rodzaje rozkazów a architektura procesora

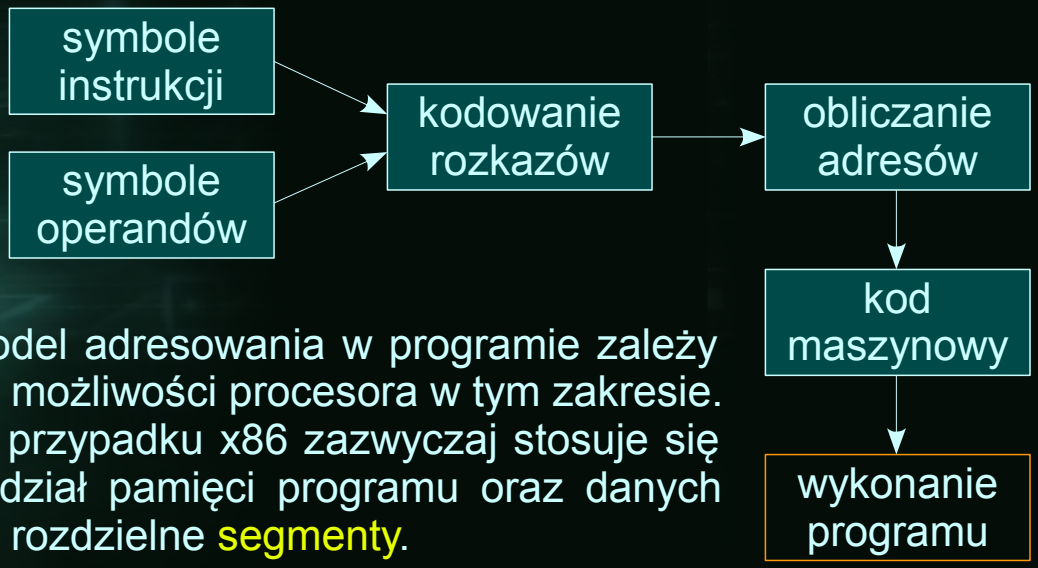
– asembler

– przykłady

Program dla procesora można wygenerować w wyniku kompilacji języka wysokiego poziomu lub przez zapis kodu maszynowego

Program:
- język asemblera
- proces kompilacji

Program zapisany w języku symbolicznym wymaga przedstawienia w formie strumienia bajtów, będących fragmentami instrukcji. Generowanie kodu maszynowego nazywa się procesem kompilacji.



Model adresowania w programie zależy od możliwości procesora w tym zakresie. W przypadku x86 zazwyczaj stosuje się podział pamięci programu oraz danych na rozdzielne segmenty.

Kod maszynowy składa się liczb, będących fragmentami rozkazów procesora. W celu ułatwienia programowania instrukcje i ich operandy mają przypisane nazwy symboliczne.

instrukcja	działanie	operandy
mov a, b	a = b	reg, mem, nat
add a, b	a = a + b	reg, mem, nat
adc a, b	a = a + b*	reg, mem, nat
sub a, b	a = a – b	reg, mem, nat
sbb a, b	a = a – b*	reg, mem, nat
and a, b	a = a & b	reg, mem, nat
or a, b	a = a | b	reg, mem, nat
xor a, b	a = a ^ b	reg, mem, nat
not a	a = ~a	reg, mem
call x	przekazanie sterowania pod x i umieszczenie adresu powrotu na stosie	
jmp x	przekazanie sterowania pod x	

* - z przeniesieniem/pożyczką.

M.Wiśniewski

Język maszynowy

rodzaje rozkazów a architektura procesora

– asembler

– przykłady

Stosując język asemblera można zaprogramować dowolne funkcjonalność lub algorytm, jednocześnie mając ogromny wpływ na jakość kodu.

Asembler x86:
- obliczenia
- odwołania do pamięci
- stos i podprogramy

Wykonać dodawanie i odejmowanie dwóch liczb 32-bitowych ze znakiem.

```
mov     eax, 100        ...
mov     ebx, -100       sub     eax, edx
add     eax, ebx        add     adx, esi
mov     edx, 200        mov     edi, edx
...
```

Wykonać dodawanie i odejmowanie dwóch liczb 64-bitowych ze znakiem.

```
mov     eax, 1122h  } liczba A
mov     ebx, 2244h  }
mov     ecx, 1010h  } liczba B
mov     edx, 2233h  }
add     eax, ecx
adc     ebx, edx
```

Wykonać mnożenie dwóch liczb 32-bitowych bez znaku.

```
mov     eax, 1001001101001111b
mov     ebx, 123
imul    ebx
```

Wynik zostanie zapisany w edx i eax

Wykonać dzielenie liczby 64-bitowej ze znakiem przez liczbę 32-bitową.

```
mov     eax, 2h
mov     edx, 7765h
mov     ecx, 1000
div     ecx
```

Wynik zostanie zapisany w eax, reszta z dzielenia w edx

M.Wiśniewski

Język maszynowy

rodzaje rozkazów a architektura procesora

– asembler

– przykłady

Stosując język asemblera można zaprogramować dowolne funkcjonalność lub algorytm, jednocześnie mając ogromny wpływ na jakość kodu.

Asembler x86:
- obliczenia
- odwołania do pamięci
- stos i podprogramy

Zapisać wynik dodawania do pamięci, następnie odczytać starszą część wyniku.

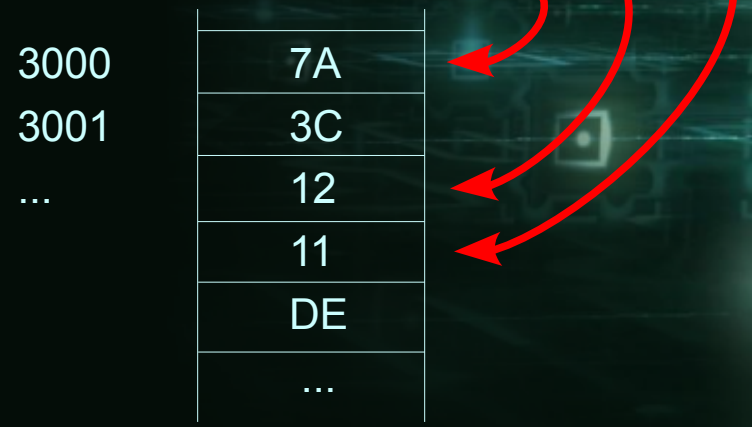
```
mov     eax, 1100
mov     ebx, 1100h
add     eax, ebx
mov     ebx, 100h
mov     [ebx], eax
mov     dx, [ebx]
mov     ax, [ebx + 2]
mov     dx, [100h]
mov     ax, [102h]
mov     di, 100h
mov     dx, [di]
add     di, 2
mov     ax, [di]
...
```

Przesłanie 2 bajtów, młodszej części.

Przesłanie 2 bajtów, starszej części.

Wykonać przesłania danych 8-bitowych z pamięci do rejestru akumulatora.

```
mov     esi, 2
mov     ebx, 3000h
mov     al, [ebx + esi + 1]
mov     al, [ebx + esi]
mov     al, [ebx]
```



Język maszynowy

rodzaje rozkazów a architektura procesora

– asembler

– przykłady

Stosując język asemblera można zaprogramować dowolne funkcjonalność lub algorytm, jednocześnie mając ogromny wpływ na jakość kodu.

Asembler x86:
- obliczenia
- odwołania do pamięci
- stos i podprogramy

Zapisać program wykonujący obliczenia, które zostały zawarte w podprogramie.

```
start:
        mov     ax, 100
        mov     bx, 200
        call    mnożenie
        xor     ax, ax
        ...
mnożenie:
        mul     bx
        mov     bx, dx
        ret
        ...
```

Wywołanie podprogramu.

Powrót z podprogramu, adres powrotu został umieszczony na stosie.

W programie wykorzystano etykiety, które w trakcie kompilacji zostaną zamienione na właściwe adresy w pamięci.

Zapisać program, który przekazuje przez stos parametry do podporgramu.

```
start:
        mov     ax, 100
        push    ax
        mov     ax, 200
        push    ax
        call    mnożenie
        mov     cx, ax
        …
mnożenie:
        push    bp
        mov     bp, sp
        mov     ax, [bp + 4]
        mul     word ptr [bp + 2]
        ret     04h
        ...
```

Projekt μP – koncepcja, model rozkazu

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

- założenia
- schemat blokowy
- projekt rozkazu

Projektowanie CPU należy rozpocząć od określenia jego przeznaczenia, funkcji jakie ma realizować oraz dostępnych zasobów konstrukcyjnych.

Założenia:

- typy danych, rozkazy
- obsługa pamięci
- urządzenia we/wy

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

- założenia
- schemat blokowy
- projekt rozkazu

Projektowanie CPU należy rozpocząć od określenia jego przeznaczenia, funkcji jakie ma realizować oraz dostępnych zasobów konstrukcyjnych.

Założenia:
- typy danych, rozkazy
- obsługa pamięci
- urządzenia we/wy

W zakresie typów danych procesor będzie posiadał następujące cechy:
- długość słowa maszynowego: 8 bitów,
- dostępne będą dane o długości 1 bajta,
- dostępne będą dane o długości 2 bajtów (tylko w pewnych operacjach),
- obsługiwane będą liczby bez znaku oraz liczby ze znakiem zapisane w kodzie U2.

Procesor będzie posiadał następujące grupy rozkazów:
- arytmetyczne: +, -, *,
- logiczne: not, and, or, xor,
- obsługa pętli,
- realizujące rozgałęzienia programu z rozróżnieniem typu wyniku (ze znakiem / bez znaku),
- zapis słowa do urządzenia wyjściowego,
- obsługa stosu,
- wywołanie podprogramu,
- powrót z podprogramu,
- skok międzysegmentowy.

M.Wiśniewski

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

– założenia

– schemat blokowy

– projekt rozkazu

Projektowanie CPU należy rozpocząć od określenia jego przeznaczenia, funkcji jakie ma realizować oraz dostępnych zasobów konstrukcyjnych.

Założenia:

- typy danych, rozkazy
- obsługa pamięci
- urządzenia we/wy

W zakresie typów danych procesor będzie posiadał następujące cechy:

- długość słowa maszynowego: 8 bitów,
- dostępne będą dane o długości 1 bajta,
- dostępne będą dane o długości 2 bajtów (tylko w pewnych operacjach),
- obsługiwane będą liczby bez znaku oraz liczby ze znakiem zapisane w kodzie U2.

Procesor będzie posiadał następujące grupy rozkazów:

- arytmetyczne: +, -, *,
- logiczne: not, and, or, xor,
- obsługa pętli,
- realizujące rozgałęzienia programu z rozróżnieniem typu wyniku (ze znakiem / bez znaku),
- zapis słowa do urządzenia wyjściowego,
- obsługa stosu,
- wywołanie podprogramu,
- powrót z podprogramu,
- skok międzysegmentowy.

Procesor będzie posiadał mechanizm segmentacji, przy czym długość słowa adresowego będzie wynosiła 20 bitów. Ponadto dostępne będą tryby adresowania pamięci operacyjnej:

- bezpośredni – w postaci selektor:[offset],
- pośredni rejestrowy (opcjonalnie),
- pośredni rejestrowy z przemieszczeniem.

Przewiduje się również obsługę stosu programowego, w postaci kolejki LIFO, umożliwiającego obsługę zmiennych lokalnych podprogramów.

M.Wiśniewski

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

– założenia

– schemat blokowy

– projekt rozkazu

Projektowanie CPU należy rozpocząć od określenia jego przeznaczenia, funkcji jakie ma realizować oraz dostępnych zasobów konstrukcyjnych.

Założenia:

- typy danych, rozkazy
- obsługa pamięci
- urządzenia we/wy

W zakresie typów danych procesor będzie posiadał następujące cechy:

- długość słowa maszynowego: 8 bitów,
- dostępne będą dane o długości 1 bajta,
- dostępne będą dane o długości 2 bajtów (tylko w pewnych operacjach),
- obsługiwane będą liczby bez znaku oraz liczby ze znakiem zapisane w kodzie U2.

Współpraca z urządzeniami we/wy będzie obejmowała obsługę jednego przerwania sprzętowego oraz możliwość zapisu/odczytu bajtu lub słowa do/z urządzenia.

Adres urządzenia będzie wystawiany na magistrali adresowej procesora.

Procesor będzie posiadał mechanizm segmentacji, przy czym długość słowa adresowego będzie wynosiła 20 bitów. Ponadto dostępne będą tryby adresowania pamięci operacyjnej:

- bezpośredni – w postaci selektor:[offset],
- pośredni rejestrowy (opcjonalnie),
- pośredni rejestrowy z przemieszczeniem.

Przewiduje się również obsługę stosu programowego, w postaci kolejki LIFO, umożliwiającego obsługę zmiennych lokalnych podprogramów.

Procesor będzie posiadał następujące grupy rozkazów:

- arytmetyczne: +, -, *,
- logiczne: not, and, or, xor,
- obsługa pętli,
- realizujące rozgałęzienia programu z rozróżnieniem typu wyniku (ze znakiem / bez znaku),
- zapis słowa do urządzenia wyjściowego,
- obsługa stosu,
- wywołanie podprogramu,
- powrót z podprogramu,
- skok międzysegmentowy.

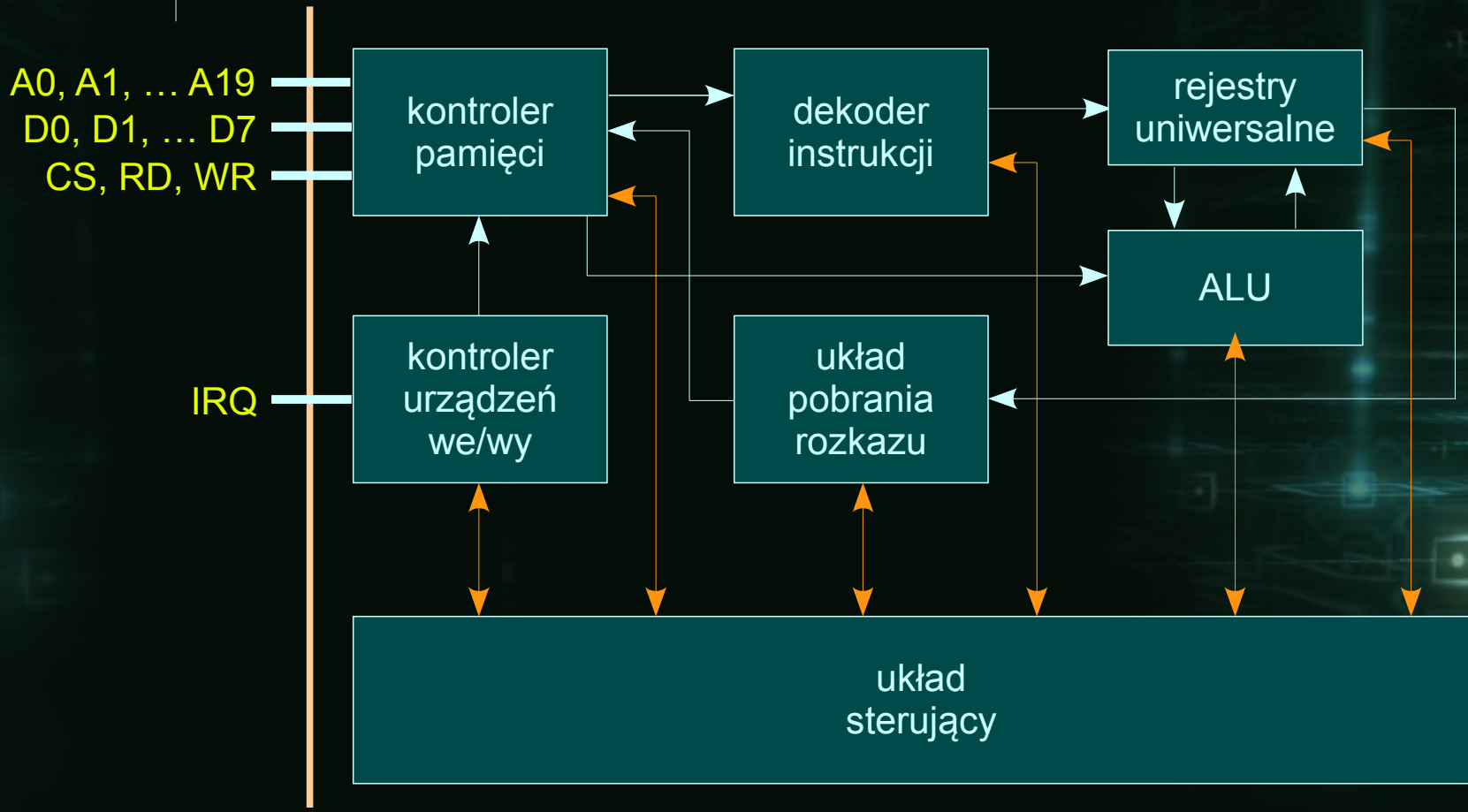
Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

- założenia
- schemat blokowy
- projekt rozkazu

Schemat blokowy przedstawia prawdopodobny podział projektowanego CPU na moduły, które będą wymagane do implementacji procesora. W trakcie wykładu model procesora będzie ewaluował, aż do osiągnięcia końcowej, przedprodukcyjnej formy.

Schemat procesora:



Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

- założenia
- schemat blokowy
- projekt rozkazu

Podczas projektowania rozkazu należy wziąć pod uwagę rodzaje rozkazów, typy argumentów i tryby pracy procesora.

Etapy:
- operacje
- model rozkazu

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

- założenia
- schemat blokowy
- projekt rozkazu

Podczas projektowania rozkazu należy wziąć pod uwagę rodzaje rozkazów, typy argumentów i tryby pracy procesora.

Etapy:
- operacje
- model rozkazu

Operacje wykonywane przez procesor:
- jednoargumentowe:
  - push reg
  - pop reg
  - reg ← not reg
- dwuargumentowe:
  - reg ← opr
  - opr ← reg
  - reg ← reg + opr
  - reg ← reg – opr
  - reg ← reg + cf + opr
  - reg ← reg – cf – opr
  - reg ← reg * opr
  - reg ← reg and opr
  - reg ← reg or opr
  - reg ← reg xor opr
- pozostałe:
  - call nat
  - ret
  - jmp nat
  - jmp far sel:nat
  - j?? nat
  - cmp reg, opr
  - out nat, R0
  - in R0, nat
  - rep nat

gdzie:
- opr może być rejestrem innym niż reg lub adresem w pamięci typu: sel:[nat] lub sel:[reg + nat],
- ?? = o/no, s/ns, e/ne, b(c)/nb(nc), l, le, g, ge.

M.Wiśniewski

Projekt μP

projekt mikroprocesora – zadanie realizowane na wykładzie

– założenia

– schemat blokowy

– projekt rozkazu

Podczas projektowania rozkazu należy wziąć pod uwagę rodzaje rozkazów, typy argumentów i tryby pracy procesora.

Etapy:
- operacje
- model rozkazu

Operacje wykonywane przez procesor:
- jednoargumentowe:
  - push reg - pop reg
  - reg ← not reg

...

- dla operacji jednoargumentowych:
  000 MCC RR [nat8] [nat8]
  - dla M = 0 - RR – kod rejestru, CC – kod operacji dla push, pop, not i reg ← nat,
  - dla M = 1 - CC i RR tworzą kod 4 bitowy dla operacji ret, call, jmp far, in, out oraz rep.
- dla operacji dwuargumentowych:
  001 xCCCC RRSLRRxx - SL=11, opr = reg,
  001 xCCCC RRSLxxxx - SL≠11, opr = sel,
  010 0CCCC RRSLxxxx nat8 - opr = sel:[nat],
  011 0CCCC RRSLRRxx nat8 - opr = sel:[reg + nat],
  - CCCC – kod rozkazu dla: reg ← opr, +, +c, –, –c, *, or, and, xor i cmp,
  - RR – rejestr (R0 = 00, R1 = 01, R2 = 10 , R3/SP = 11)
- dla operacji przesłania do pamięci:
  10 SL RR RR nat8
  - dla SL = 11 - reg:[nat] ← reg,
  - dla SL ≠ 11 - sel:[reg + nat] ← reg.

- zapis rejestrów segmentowych:
  010 1SLRR - sel ← reg,
  011 1SLxx nat8 nat8 - sel ← nat8,nat8,
  - SL – selektor (CS = 00!, DS = 01, SS = 10).

- dla operacji skoków warunkowych:
  11 xx CCCC nat8
  gdzie CCCC – kod rodzaju warunku:

0000 – overflow,	0001 – not overflow,
0010 – sign,	0011 – not sign,
0100 – equal (zero),	0101 – not equal,
0110 – below (carry),	0111 – not below,
1000 – below or equal,	1001 – above,
1010 – less,	1011 – less or equal,
1100 – greater or equal,	1101 – greater.

Koniec wykładu