Notatki: Organizacja i Architektura Komputerów

Twoje Imie i Nazwisko

January 16, 2025

Contents

1 Reprezentacja informacji w komputerze - arytmetyka i logika

1.1 Systemy liczbowych

System binarny (dwójkowy)

Opis: W systemie binarnym używamy tylko dwóch cyfr: 0 i 1. Każda pozycja reprezentuje potege liczby 2.

Przykład obliczenia:

Liczba binarna: 1011_2

$$1 \times 2^3 = 8, \quad 0 \times 2^2 = 0, \quad 1 \times 2^1 = 2, \quad 1 \times 2^0 = 1$$

Suma: $8 + 0 + 2 + 1 = 11_{10}$

System dziesietny

Opis: Używamy dziesieciu cyfr (0-9), gdzie każda pozycja jest potega 10.

Przykład: Liczba 235_{10} oznacza:

$$2 \times 10^2 = 200, \quad 3 \times 10^1 = 30, \quad 5 \times 10^0 = 5$$

Suma: $200 + 30 + 5 = 235$

System szesnastkowy (heksadecymalny)

Opis: Używamy 16 symboli: 0–9 oraz A, B, C, D, E, F (gdzie $A = 10$, $B = 11, \ldots, F = 15$). Każda pozycja jest potega 16.

Przykład obliczenia: Liczba $1A3_{16}$

$$1 \times 16^2 = 1 \times 256 = 256, \quad A\ (10) \times 16^1 = 10 \times 16 = 160, \quad 3 \times 16^0 = 3$$

Suma: $256 + 160 + 3 = 419_{10}$

1.2 Arytmetyka binarna

Dodawanie binarne

Zasada: Podobnie jak w systemie dziesietnym, ale reguły sa nastepujace:

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 1 + 1 = 10_2 \quad \text{(zapisujemy 0, przenosimy 1)}$$

Przykład: Dodawanie $1011_2 + 1101_2$
Rozpisanie kolumnami:

$$\begin{array}{r} 1\ \ 0\ \ 1\ \ 1 \\ +\,1\ \ 1\ \ 0\ \ 1 \\ \hline 1\,1\,0\,0\,0 \end{array}$$

Objaśnienie:

- **Kolumna 0 (LSB):** $1 + 1 = 10_2 \rightarrow$ zapisujemy 0, przenosimy 1.

- **Kolumna 1:** $1 + 0 +$ przeniesienie $1 = 10_2 \rightarrow$ zapisujemy 0, przenosimy 1.
- **Kolumna 2:** $0 + 1 +$ przeniesienie $1 = 10_2 \rightarrow$ zapisujemy 0, przenosimy 1.
- **Kolumna 3:** $1 + 1 +$ przeniesienie $1 = 11_2 \rightarrow$ zapisujemy 1, przenosimy 1.
- Dodatkowy bit przeniesienia: wynik to 11000_2.

Odejmowanie binarne

Metoda pożyczkowa: Działa analogicznie do odejmowania w systemie dziesietnym. Jeśli dolny bit jest wiekszy od górnego, pożyczamy z kolejnej kolumny.

Przykład: Odejmowanie $1010_2 - 0111_2$

(Szczegółowy schemat obliczeń można przedstawić przy użyciu tabeli pożyczek dla każdej kolumny.)

1.3 Reprezentacja liczb ze znakiem

Kod uzupełnień do dwóch (2's complement)

Cel: Umożliwienie reprezentacji liczb dodatnich i ujemnych.

Metoda:

1. Zapisz wartość dodatnia w systemie binarnym.
2. Zaneguj wszystkie bity (0 na 1 i 1 na 0).
3. Dodaj 1 do uzyskanego wyniku.

Przykład: Reprezentacja liczby -5 w 8-bitowym systemie.

1. Liczba 5: $\texttt{00000101}_2$.
2. Negacja: $\texttt{11111010}_2$.
3. Dodanie 1:

$$\begin{array}{rr} & \texttt{11111010}_2 \\ + & \texttt{00000001}_2 \\ \hline & \texttt{11111011}_2 \end{array}$$

Wynik: $\texttt{11111011}_2$ reprezentuje -5.

1.4 Operacje logiczne

Podstawowe operatory logiczne

AND (koniunkcja): Wynik 1 tylko, gdy oba bity sa 1.
1 AND 1 = 1, 1 AND 0 = 0.

OR (alternatywa): Wynik 1 gdy przynajmniej jeden z bitów jest 1.
1 OR 0 = 1, 0 OR 0 = 0.

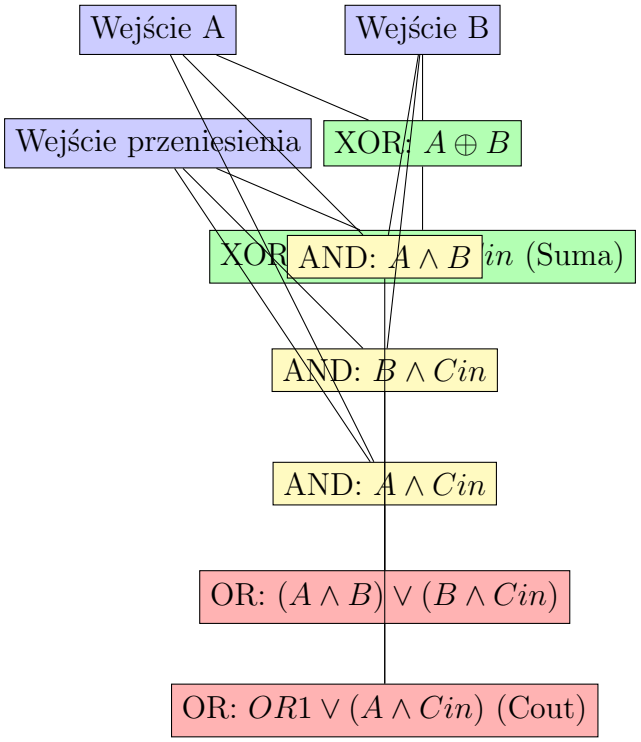
NOT (negacja): Odwraca wartość bitu.
NOT $1 = 0$, NOT $0 = 1$.

XOR (alternatywa wykluczajaca): Wynik 1, jeżeli bity sa różne.
1 XOR $0 = 1$, 1 XOR $1 = 0$.

Schemat operacji logicznych

A	B	A AND B	A OR B	A XOR B	NOT A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Schemat blokowy sumatora bitowego



Przykładowe pytania – Reprezentacja informacji

1. **Pytanie:** Jaka wartość w systemie dziesietnym ma liczba binarna 1101_2?
   Odpowiedź:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13_{10}$$

2. **Pytanie:** Opisz krok po kroku, jak uzyskać reprezentacje liczby -5 w 8-bitowym systemie za pomoca kodu uzupełnień do dwóch.
   Odpowiedź:

   - Zapisz 5 w postaci 8-bitowej: $\texttt{00000101}_2$.
   - Zaneguj wszystkie bity: $\texttt{11111010}_2$.

- Dodaj 1: $\texttt{11111010}_2 + 1_2 = \texttt{11111011}_2$.

3. **Pytanie:** Co daje operacja XOR oraz podaj przykład działania dla bitów 1 i 0.
   Odpowiedź: Operacja XOR daje wartość 1 wtedy, gdy bity sa różne, czyli:

$$1 \text{ XOR } 0 = 1.$$

2 Układy kombinacyjne i sekwencyjne (analiza działania i projektowanie)

2.1 Układy kombinacyjne

Definicja

Opis: Układy kombinacyjne to układy, których wyjścia zależa wyłacznie od aktualnych stanów wejść. Nie posiadaja elementu pamieci, wiec przy zmianie wejść natychmiast nastepuje zmiana wyjść.

Projektowanie układów kombinacyjnych

1. **Tabela prawdy:** Sporzadzamy tabele określajaca zależność miedzy wejściami a wyjściami.
2. **Minimalizacja funkcji logicznych:** Używamy metod, takich jak mapa Karnaugha.
3. **Implementacja:** Wyrażenia logiczne implementujemy przy użyciu bramek (AND, OR, NOT, XOR, ...).

Przykłady układów kombinacyjnych

Pełny sumator (Full Adder) **Wejścia:** A, B, C_{in} (przeniesienie wejściowe)

Wyjścia: S (suma), C_{out} (przeniesienie wyjściowe)

Równania logiczne:

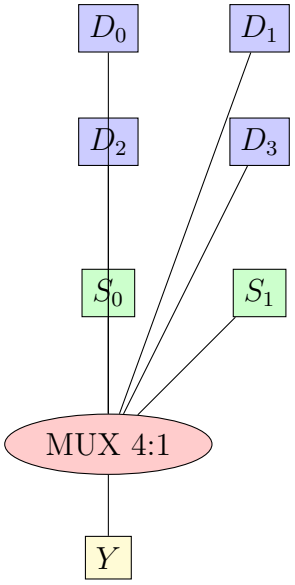
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = (A \cdot B) + (B \cdot C_{in}) + (A \cdot C_{in})$$

Multiplekser (MUX) **Opis:** Układ wybierajacy jeden z wielu sygnałów wejściowych w oparciu o sygnały sterujace.

Przykład: Multiplekser 4:1, który ma 4 wejścia D_0, D_1, D_2, D_3 oraz 2 sygnały wyboru S_0, S_1. Wyjście Y odpowiada jednemu z wejść w zależności od wartości S_0 i S_1.

Schemat Multipleksera 4:1:



2.2 Układy sekwencyjne

Definicja

Opis: Układy sekwencyjne to te, które oprócz bieżacych stanów wejść, uwzgledniaja poprzednie stany, czyli posiadaja element pamieci.

Elementy podstawowe

Przerzutniki: Podstawowe elementy pamieci przechowujace jeden bit informacji.

- **RS (Reset-Set):** Prosty w budowie, ale podatny na niejednoznaczności.
- **D (Data):** Wartość wejścia jest zapisywana przy zboczu sygnału zegarowego.
- **JK:** Ulepszona wersja przerzutnika RS.
- **T (Toggle):** Zmienia stan przy aktywnym sygnale zegara; stosowany w licznikach.

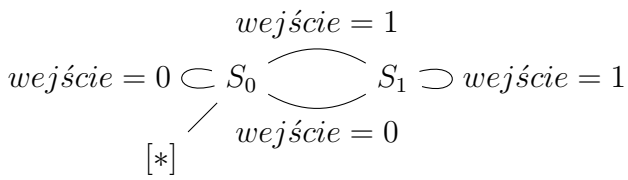
Przykłady układów sekwencyjnych

- **Licznik binarny:** Układ, który cyklicznie zwieksza wartość, zbudowany na przerzutnikach (T lub D).
- **Rejestr:** Przechowuje dane w systemie komputerowym, może być równoległy.

Projektowanie układów sekwencyjnych

1. Określ liczbe stanów – sporzadź diagram stanów lub tabele przejść.
2. Określ sygnały sterujace i warunki przejść.
3. Zaimplementuj logike przy użyciu bramek i przerzutników.

Przykładowy diagram stanów (automat sekwencyjny)



Objaśnienie: Diagram przedstawia dwa stany (S_0 i S_1) oraz przejścia zależne od wartości sygnału wejściowego.

Przykładowe pytania – Układy kombinacyjne i sekwencyjne

1. **Pytanie:** Co odróżnia układy kombinacyjne od sekwencyjnych?
   Odpowiedź: Układy kombinacyjne zależa wyłacznie od bieżacych stanów wejść, natomiast sekwencyjne uwzgledniaja również poprzednie stany (posiadaja pamieć).

2. **Pytanie:** Wyprowadź równania logiczne dla pełnego sumatora.
   Odpowiedź:

$$S = A \oplus B \oplus C_{in} \quad \text{oraz} \quad C_{out} = (A \cdot B) + (B \cdot C_{in}) + (A \cdot C_{in})$$

3. **Pytanie:** Jakie typy przerzutników spotykamy w układach sekwencyjnych?
   Odpowiedź: Możemy spotkać: RS, D, JK oraz T. Przykładowe zastosowania to rejestry, liczniki i automaty stanów.

3 Jednostka centralna i pamieć

3.1 Jednostka centralna (CPU)

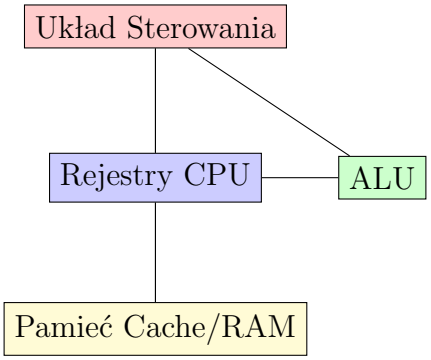
Główne składniki CPU

- **ALU (Arithmetic Logic Unit):** Wykonuje operacje arytmetyczne oraz logiczne.
- **Rejestry:** Małe, bardzo szybkie obszary pamieci w CPU (np. akumulator, licznik programu, wskaźnik stosu).
- **Układ sterowania (Control Unit):** Pobiera instrukcje z pamieci, dekoduje je i steruje przepływem danych pomiedzy komponentami.

Cykl maszynowy (Fetch-Decode-Execute)

1. **Fetch (Pobranie):** Instrukcja jest pobierana z pamieci wg adresu określonego przez licznik programu.
2. **Decode (Dekodowanie):** Instrukcja jest interpretowana przez dekoder.
3. **Execute (Wykonanie):** Instrukcja jest wykonywana (operacja arytmetyczna, logiczna lub operacja na pamieci).
4. **Write-back (Zapis):** Wynik operacji zapisywany jest do rejestru lub pamieci.

Schemat blokowy CPU i pamieci



Objaśnienie: Układ sterowania koordynuje przepływ danych miedzy rejestrami, ALU i pamiecia.

3.2 Pamieć

Hierarchia pamieci

Rejestry: Najszybsze, bardzo małe.

Cache: Szybka pamieć podreczna (np. L1, L2, L3).

Pamieć główna (RAM): Przechowuje dane i programy podczas pracy komputera.

Pamieć masowa (HDD, SSD): Najwieksza pojemność, wolniejszy dostep.

Rodzaje pamieci

- **RAM (Random Access Memory):** DRAM lub SRAM, dane tracone przy wyłaczeniu zasilania.
- **ROM (Read Only Memory):** Pamieć stała, zawierajaca program rozruchowy.

Adresowanie pamieci

- **Adres liniowy:** Cała pamieć jako jeden obszar adresowy.
- **Adresowanie segmentowe:** Pamieć podzielona na segmenty (kod, dane, stos).

Przykładowe pytania – Jednostka centralna i pamieć

1. **Pytanie:** Wymień trzy główne składniki CPU i opisz ich funkcje.
   Odpowiedź: ALU (operacje arytmetyczne i logiczne), rejestry (przechowywanie danych i adresów) oraz układ sterowania (koordynacja cyklu maszynowego).
2. **Pytanie:** Co to jest cykl maszynowy CPU i jakie sa jego etapy?
   Odpowiedź: Fetch, Decode, Execute oraz Write-back.
3. **Pytanie:** Wyjaśnij hierarchie pamieci w komputerze.
   Odpowiedź: Rejestry (najszybsze, najmniejsze), cache (szybka, mała), RAM (średnia szybkość i pojemność) oraz pamieć masowa (duża pojemność, wolniejsza).

4 Organizacja równoległa

4.1 Podstawowe pojecia i cele

Cel organizacji równoległej: Zwiekszenie wydajności systemu przez jednoczesne wykonywanie wielu operacji.

Dlaczego równoległość? W miare rozwoju mocy pojedynczych procesorów pojawiaja sie ograniczenia (prawo Amdahla). Równoległość pozwala wykorzystać wiele jednostek obliczeniowych jednocześnie.

4.2 Poziomy równoległości

Równoległość na poziomie bitów: **Opis:** Operacje na wielu bitach jednocześnie. Przykładem sa operacje SIMD (Single Instruction, Multiple Data) – jedna instrukcja przetwarza wiele danych równocześnie.

Równoległość na poziomie instrukcji: **Opis:** W ramach jednego procesora różne instrukcje wykonywane sa równolegle (pipelining).
Przykład: Kolejne etapy potoku instrukcji (fetch, decode, execute) dla różnych instrukcji.

Równoległość na poziomie procesorów (MIMD): **Opis:** Systemy wieloprocesorowe, gdzie każdy procesor wykonuje inne instrukcje na różnych danych.
Przykład: Serwery wieloprocesorowe, klastry obliczeniowe.

4.3 Architektury równoległe

SIMD (Single Instruction, Multiple Data): **Opis:** Ta sama instrukcja wykonywana równocześnie na zestawie danych.
Przykład: Procesory graficzne (GPU).

MIMD (Multiple Instruction, Multiple Data): **Opis:** Każdy procesor wykonuje własny zestaw instrukcji.
Przykład: Systemy klastrowe i wielordzeniowe.

4.4 Implementacja i wyzwania

Systemy wielowatkowe

Opis: Podział programu na watki wykonywane równolegle przez różne rdzenie procesora.
Zaleta: Lepsze wykorzystanie zasobów i skrócenie czasu wykonania.

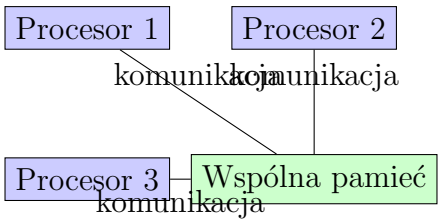
Mechanizmy komunikacji

- **Pamieć wspólna:** Wszystkie procesory maja dostep do tej samej przestrzeni adresowej.
- **Przesyłanie komunikatów:** Procesory wymieniaja dane przez wysyłanie i odbieranie komunikatów, typowe w systemach rozproszonych.

Wyzwania

- **Synchronizacja:** Użycie semaforów, muteksów, barier synchronizacyjnych.
- **Skalowalność:** Zwiekszanie liczby jednostek obliczeniowych bez nadmiernych narzutów komunikacyjnych.
- **Złożoność projektowania:** Efektywne algorytmy równoległe wymagaja dodatkowych technik i optymalizacji.

Schemat organizacji równoległej (system wieloprocesorowy)



Objaśnienie: Każdy procesor komunikuje sie z centralna, wspólna pamiecia.

Przykładowe pytania – Organizacja równoległa

1. **Pytanie:** Czym różni sie architektura SIMD od MIMD?
   Odpowiedź:
   • **SIMD:** Jedna instrukcja wykonywana na wielu danych jednocześnie (np. GPU).
   • **MIMD:** Każdy procesor wykonuje własny zestaw instrukcji (systemy klastrowe).
2. **Pytanie:** Jakie mechanizmy synchronizacji sa wykorzystywane w systemach równoległych?
   Odpowiedź: Semafory, muteksy i bariery synchronizacyjne.
3. **Pytanie:** Wymień zalety i wyzwania organizacji równoległej.
   Odpowiedź:
   • **Zalety:** Zwiekszenie wydajności przez równoległe wykonywanie zadań, lepsze wykorzystanie zasobów.
   • **Wyzwania:** Synchronizacja, komunikacja miedzy procesorami, skalowalność oraz złożoność projektowania.

5 Podsumowanie

W niniejszych notatkach omówiliśmy:

1. **Reprezentacje informacji:**
   • Systemy liczbowe (binarny, dziesietny, szesnastkowy).
   • Arytmetyke binarna (dodawanie i odejmowanie).
   • Reprezentacje liczb ze znakiem w kodzie uzupełnień do dwóch.
   • Operacje logiczne i schematy bramek.
2. **Układy kombinacyjne i sekwencyjne:**
   • Definicje, projektowanie i przykłady (pełny sumator, multiplekser, liczniki, rejestry, automaty stanów).

3. **Jednostke centralna i pamieć:**
   • Składniki CPU, cykl maszynowy oraz hierarchia pamieci.

4. **Organizacje równoległa:**
   • Poziomy równoległości, architektury SIMD i MIMD, systemy wielowatkowe i wyzwania.

Te szczegółowe notatki wraz z ilustracjami, schematami i przykładowymi pytaniami powinny pomóc w lepszym zrozumieniu zagadnień z zakresu organizacji i architektury komputerów.

Powodzenia w nauce!