CZĘŚĆ I Test jednokrotnego wyboru. Za zaznaczenie w zadaniu prawidłowej odpowiedzi otrzymuje się 1 punkt

1. Architektura procesora cechująca się znaczną liczbą złożonych rozkazów to:
   1. architektura potokowa

# architektura CISC

* 1. architektura superskalarna
  2. architektura RISC

1. Macierz RAID 5 zbudowana z trzech dysków o pojemności 1 TB każdy, jest widziana przez system operacyjny jako dysk logiczny o pojemności
   1. 3 TB

# 2 TB

* 1. 1 TB
  2. 1.5 TB

1. Minimalna liczba dysków potrzebnych do zbudowania macierzy typu RAID 1 wynosi:
   1. 5
   2. 3
   3. 2
   4. 4
2. Wykonywanie przez procesor operacji XOR na liczbach całkowitych jest realizowany w:
   1. dowolnym rejestrze ogólnego przeznaczenia
   2. jednostce FPU
   3. dekoderze rozkazów

# jednostce ALU

1. Interfejs RS-232C jest standardem przesyłania danych wykorzystującym:
   1. synchroniczne łącze równoległe
   2. synchroniczne łącze szeregowy
   3. asynchroniczne łącze równoległe

# asynchroniczne łącze szeregowe

1. Uszereguj fazy wykonywania rozkazu w kolejności od realizowanej najwcześniej do realizowanej najpóźniej:
   1. pobieranie argumentów- pobieranie rozkazu- wykonywanie rozkazu- zapisywanie wyników

# pobieranie rozkazu- dekodowanie rozkazu- wykonywanie rozkazu- zapisywanie wyników

* 1. pobieranie rozkazu- wykonywanie rozkazu- pobieranie argumentów- zapisywanie wyników
  2. dekodowanie rozkazu- pobieranie rozkazu- wykonywanie rozkazu- zapisywanie wyników

CZĘŚĆ II Test wielokrotnego wyboru. Za zaznaczenie w zadaniu wszystkich prawidłowych odpowiedzi i jednoczesne niezaznaczenie nieprawidłowych odpowiedzi otrzymuje się 2 punkty.

1. Które, z podanych cech dotyczą architektury RISC:

# nieduży zestaw stosunkowo prostych rozkazów o stałej długości

* 1. duża liczba złożonych trybów adresowania

# duża liczba rejestrów wewnętrznych ogólnego przeznaczenia

# wykonywanie większości rozkazów w jednym cyklu zegara

1. Zadaniem jednostki adresowania procesora jest:
   1. pobieranie rozkazów i umieszczanie ich w podręcznej kolejce rozkazów

# wytworzenie na magistrali adresowej adresu, pod którym znajdują się pobierane rozkazy lub wymienione z procesorem dane

* 1. ustalenie operacji do wykonania przez jednostkę wykonawczą procesora
  2. pobieranie z magistrali adresowej adresów kolejnych argumentów rozkazu

1. Które z poniższych interfejsów cechuje szeregowa transmisja danych:

# FireWire

# eSATA

* 1. IEEE-1284
  2. SCSI

c jest równoległe a d szeregowe

1. Pamięć typu DDR/DDR2/DDR3

# jest pamięcią typu SDRAM umożliwiającą przesyłanie w jednym cyklu zegarowym dwóch słów danych

* 1. jest pamięcią półprzewodnikową o dostępie bezpośrednim stosowaną w komputerach jako pamięć operacyjna

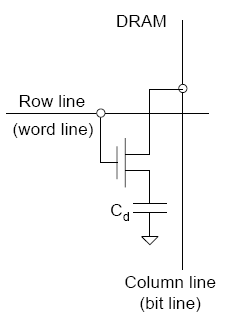
# jest pamięcią dynamiczną posiadającą autonomiczny układ odświeżania zawartości

# jest pamięcią dynamiczną o dostępie swobodnym przystosowaną do pakietowego (burst) przesyłania danych

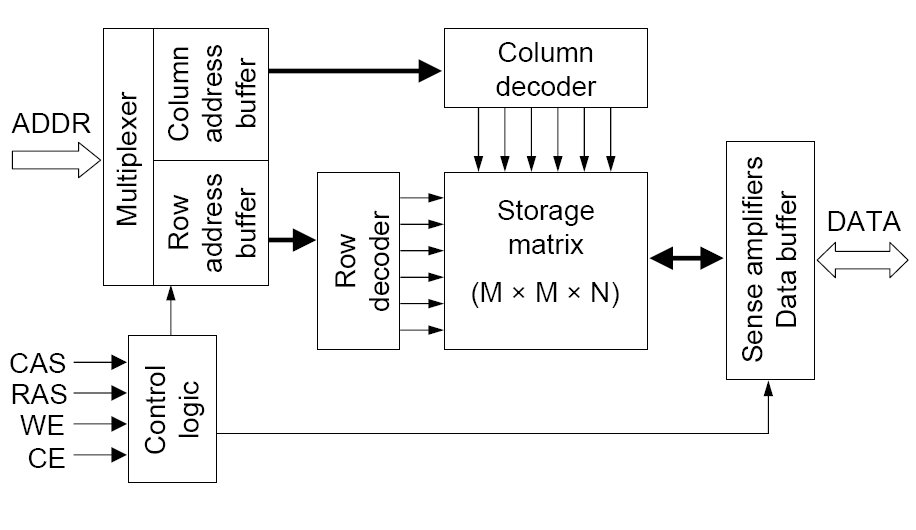
CZĘŚĆ III Zadania otwarte. Za każde z zadań można otrzymać 5 punktów

1. Przedstaw budowę wewnętrzną pamięci SDRAM

Podstawową komórką pamięci SDRAM jest, tak samo jak w pamięci DRAM, układ złożony z tranzystora i kondensatora. Układ ten przechowuje ładunek elektryczny, przez co komórki wymagają okresowego odświeżania ładunków:



Organizacja wewnętrzna pamięci DRAM:



Układ pamięci SDRAM zbudowany jest z niezależnych od siebie banków (najczęściej dwóch lub czterech), które adresowane są naprzemiennie, co przyspiesza proces dostępu do danych. W układ pamięci SDRAM wbudowany jest licznik odświeżania i układy logiczne, które pozwalają na jego autonomiczne odświeżanie

1. Opisz ideę działania pamięci podręcznej wykorzystującej mechanizm odwzorowania sekcyjno-skojarzeniowego

Odwzorowanie sekcyjno-skojarzeniowe jest metodą, która łączy zalety odwzorowania bezpośredniego i skojarzeniowego. Cechą charakterystyczną tego typu podejścia jest podział pamięci podręcznej na 2, 4 lub 8 sekcji (2- Way-Set , 4-Way-Set , 8-Way-Set ) o rozmiarze równym wielkości strony pamięci operacyjnej. Ponieważ pamięć operacyjna odwzorowywana jest na każdą sekcję w sposób bezpośredni, wyszukanie odpowiedniej informacji w pamięci podręcznej wymaga przeprowadzenia co najwyżej 8 porównań (jedno porównanie na sekcję). Cechą odwzorowania skojarzeniowego i sekcyjno-skojarzeniowego jest konieczność wyboru tego wiersza pamięci podręcznej, który zostanie zastąpiony przez nowo ładowany wiersz z pamięci operacyjnej. W tym celu kontroler pamięci podręcznej implementuje jeden z następujących algorytmów: FIFO, LFU (Least Frequently Used), LRU (Least Re

1. Omów podstawowe elementy i założenia architektury potokowej

Potokowość (ang. pipelining) — technika budowy procesorów polegająca na podziale logiki

procesora odpowiedzialnej za proces wykonywania programu (instrukcji procesora) na

specjalizowane grupy w taki sposób, aby każda z grup wykonywała część pracy związanej z

wykonaniem rozkazu. Grupy te są połączone sekwencyjnie — potok (ang. pipe) — i

wykonują pracę równocześnie, pobierając dane od poprzedniego elementu w sekwencji. W

każdej z tych grup rozkaz jest na innym stadium wykonania. Można to porównać do taśmy

produkcyjnej. W uproszczeniu, potok wykonania instrukcji procesora może wyglądać

następująco: Uproszczony schemat potokowości. Części rozkazów oznaczone na zielono

wykonywane są równocześnie. Uproszczony schemat potokowości. Części rozkazów

oznaczone na zielono wykonywane są równocześnie.

· Pobranie instrukcji z pamięci - ang. instruction fetch (IF)

· Zdekodowanie instrukcji - ang. instruction decode (ID)

· Wykonanie instrukcji - ang. execute (EX)

· Dostęp do pamięci - ang. memory access (MEM)

· Zapisanie wyników działania instrukcji - ang. store; write back (WB)

W powyższym 5-stopniowym potoku, przejście przez wszystkie stopnie potoku (wykonanie

jednej instrukcji) zabiera co najmniej 5 cykli zegarowych. Jednak ze względu na jednoczesną

pracę wszystkich stopni potoku, jednocześnie wykonywanych jest 5 rozkazów procesora,

każdy w innym stadium wykonania. Oznacza to, że taki procesor w każdym cyklu zegara

rozpoczyna i kończy wykonanie jednej instrukcji i statystycznie wykonuje rozkaz w jednym

cyklu zegara. Każdy ze stopni potoku wykonuje mniej pracy w porównaniu do pojedynczej

logiki, dzięki czemu może wykonać ją szybciej - z większą częstotliwością - tak więc

dodatkowe zwiększenie liczby stopni umożliwia osiągnięcie coraz wyższych częstotliwości

pracy.

Podstawowym mankamentem techniki potoku są rozkazy skoku, powodujące w najgorszym

wypadku potrzebę przeczyszczenia całego potoku i wycofania rozkazów, które następowały

zaraz po instrukcji skoku i rozpoczęcie zapełniania potoku od początku od adresu, do

którego następował skok. Taki rozkaz skoku może powodować ogromne opóźnienia w

wykonywaniu programu - tym większe, im większa jest długość potoku. Dodatkowo szacuje

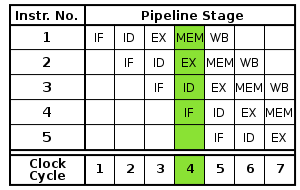
się, że dla modelu programowego x86 taki skok występuje co kilkanaście rozkazów. Z tego

powodu niektóre architektury programowe (np. S PARC) zakładały zawsze wykonanie

jednego lub większej ilości rozkazów następujących po rozkazie skoku, tzw. skok opóźniony.

Stosuje się także skomplikowane metody predykcji skoku lub metody programowania bez

użycia skoków.



– Krócej:

W cyklu rozkazowym procesora można wyróżnić cztery fazy:

· Pobrania rozkazu – Fetch, w której zachodzi pobranie rozkazu z podręcznej kolejki

rozkazów Prefetch i jeżeli istnieje taka potrzeba uzupełnienie tej kolejki

· Dekodowania rozkazu – Decode, w której określane są operacje do wykonania oraz

obliczane adresy efektywne argumentów o ile istnieją

· Wykonania – Execute – z pamięci pobierane są argumenty rozkazów i wykopnywane

są operacje ustalone w fazie Decode

· Zapisania wyników – Retire, w której następuje zapisanie wyników w miejscu

określonym w kodzie rozkazu oraz ustawienie znaczników rejestru statusowego

Każda instrukcja przechodzi przez powyższe fazy co przyspiesza wykonywanie. Sposób

taki można porównać przez analogię do taśmy produkcyjnej. Dobra organizacja pracy i

podział na różne fazy przyspiesza produkcję.

1. Napisz program (a assemblerze mikrokontrolera 8051) który po naciśnięciu przycisku podłączonego do portu P3.2 uruchomi przerwanie programujące T0 na zliczanie max liczby impulsów, zapali diodę LED P3.4. Po wygenerowaniu 20 przerwań od T0, zostanie on zatrzymany a dioda wyłączona.

$MOD842

LED EQU P3.4

ORG 000h

JMP MAIN

ORG 0003h

JMP START

ORG 00Bh

JMP FIN ; przepełnienie T0

ORG 060h

MAIN:

CLR LED ;gasimy światełko

SETB EA ;odblokowanie przerwan

SETB EX0

SETB ET0

MOV TMOD, 00001110b ; timer 0 jako counter 8 bitowy z reloadem

MOV TH0, 0EBh ; max minus 20 liczy 20 sygnałow do przepełnienia

JMP $

START:

SETB LED ; zapalamy światełko

CLR EX0 ; nie chcemy obsługiwać INT0

RETI

FIN:

CLR LED ; gasimy

SETB EX0 ; pozwalamy na start jeszcze raz

RETI

END.

$MOD842

LED EQU P3.4

ORG 0000h

JMP MAIN

ORG 0003h

JMP INT

ORG 000Bh

JMP INT\_T0

ORG 0060h

MAIN:

MOV R0, #20 ; licznik count down

MOV R1, #1

SETB EA

SETB EX0

SETB ET0

JMP $

INT:

MOV A, R1

JZ INT\_1

SUB R1, #1

MOV TMOD, #05h

MOV TCON, #10h

CPL LED ; switch led

INT\_1:

RETI

INT\_T0:

MOV A,R0

JZ KONIEC

SUB R0,#1

RETI

KONIEC:

CMP LED

CLR ET0

RETI

CZĘŚĆ IV Analiza programu. Za zadanie można otrzymać 0-3 punktów.

$MOD842

LED EQU P3.4

ERROR EQU F0 ; the 'ERROR' flag

ORG 0000h

JMP MAIN

ORG 0003h

SETB ERROR

RETI

ORG 0060h

MAIN:

JB WDS, WDRESET

MOV R0, #05

JMP START

WDRESET:

MOV R0, #025

START:

SETB IT0

SETB EX0

SETB EA

CLR EA

SETB WDWR

MOV WDCON, #72h ; Enable Watchdog timer to cause-enable WDIR bit to generate a reset and not an interrupt

SETB EA

CLR ERROR ; simulate error free operation

FLASH:

MOV A, R0

CALL DELAY

CPL LED

CLR EA

SETB WDWR

SETB WDE

SETB EA

JNB ERROR, FLASH

CLR LED

JMP $

DELAY: ; Delays by 50ms \* A, 50mSec based on 4.194304MHZ Core Clock

MOV R1,A ; Acc holds delay variable (1 clock)

DLY0: MOV R2,#01Bh ; Set up delay loop0 (2 clocks)

DLY1: MOV R3,#0FFh ; Set up delay loop1 (2 clocks)

DJNZ R3,$ ; Dec R3 & Jump here until R3 is 0 (3 clocks)

DJNZ R2,DLY1 ; Dec R2 & Jump DLY1 until R2 is 0 (3clocks)

DJNZ R1,DLY0 ; Dec R1 & Jump DLY0 until R1 is 0 (3 clocks)

RET ; Return from subroutine

END

1. Które ze zdań odpowiadają zadaniom realizowanym przez powyższy program:
   1. Dioda LED mruga z większą częstotliwością od startu programu do przejścia przerwania INT0
   2. Dioda LED mruga z mniejszą częstotliwością od momentu przyjścia przerwania INT0 do resetu systemu licznikiem watchdog// moim zdaniem nie mruga wgl
   3. Dioda LED mruga z mniejszą częstotliwością po zliczeniu 2s przez licznik watchdog
   4. Dioda LED przestaje mrugać gdy pojawi się sygnał przerwania INT0
   5. Dioda LED mruga z większą częstotliwością po restarcie systemu licznikiem watchdog

Dodatek:

Link do siriusa z powyższym programem i z dodatkowymi komentarzami które pomagają zrozumieć działanie programu:

<http://sirius.cs.put.poznan.pl/~inf94358/auth/materialy/rekurencja/ADuC842/842%20asm/WDTimer/WDtimer.asm>

**2015**

Otwarte zadanie do przedstawienia idea przetwarzania potokowego przy 8 rozkazach i podziale na 5 etapów.

A. O ile skróci się czas w stosunku do zwykłej architektury.

B. Jak zmieni się czas gdy 5 rozkaz będzie rozkazem skoku.

Opisać dlaczego i narysować.

Przyjmujemy, że każdy etap jest jednym cyklem maszynowym i trwa równo jeden cykl zegarowy.

Dodatkowo trzeba założyć jakąś liczbę potoków. Niech będzie 8, po jednym potoku na każdy rozkaz.

ad. A

czas normalny:

tn = 8 \* 5 cykli zegarowych = 40 cykli zegarowy

optymistyczny czas skrócony na potokach (przy założeniu braku skoków i odgałęzień):

ts = 5 + (8 - 1) = 12

tn - ts = 40 - 12 = 28

ad. B



t = [5 + (5 - 1)] + [5 + (3 - 1)] = 16

// Po co wykonywać od nowa całość, nie można po prostu unieważnić rozkazów, które są po skoku? Chyba, że nie zrozumiałem zapisu i to to samo, co poniżej… ale jak dla mnie skok = nie do rozkazu 6, tylko do X (bo co to za skok do kolejnej instrukcji?)

// Zapis zrobiony na szybko i przyznaję, że nie jasny,

