Architektura komputerów

Opracował: Grzegorz Gliński

Cyfrowe układy logiczne

28-02-2008

Bramki logiczne stanowią podstawowe bloki, z których zbudowany jest komputer cyfrowy. Stan wyjścia bramki logicznej zależy od elektronicznych poziomów logicznych sygnałów dostarczonych na wejście. Dla dwustronnych urządzeń logicznych poziomy logiczne są opisywane, jako prawda(1) fałsz(0) (true of fals) Alternatywnym dla takiego opisu t/f jest stan niski i wysoki (hight/ and low) Może być też reprezentowany przez włączony wyłączony

Do tworzenia układów cyfrowych wystarczy użyć kilka podstawowych bramek, z których każda wykonuje pewną funkcje logiczna. Podstawowe to są: And, Or, Not. Stany logiczne wyjść i wejść bramek SA opisywane zmiennymi boolowskimi, które przybierają wartość 0 albo 1.

Układy logiczne możemy możemy podzielić na dwa typy: układy sekwencyjne oraz układy kombinacyjne.

Układy kombinacyjne są to takie układy w których stan wyjściowy jest zależnty tylko i wyłącznie od stanu na wejściu układu. Do układów kombinacyjnych możemy zaliczyć:

- Bramki logiczne
- Sumatory
- Półsumatory
- Kodery
- Multipleksery
- Demultipleksery
- Dekodery

Układy sekwencyjne są to takie układy w których stan wyjściowy zależy od stanu na wejściu układu oraz od poprzedinego stanu układu. Do układów sekwencyjnych możemy zaliczyć:

- Przerzutniki
- Rejestry
- Liczniki
- Układy pamięci

Układy sekwencyjne możemy podzielić na układy **synchroniczne** oraz **asynchroniczne**.

W układach asynchronicznych następuje przejście z jednego stanu do drugiego bezpośrednio po zmianie stanu na wejściach.

W układach asynchronicznych zmiana stanu odbywa się w chwili wyznaczonych zmiana sygnału synchronizującego.

Poszczególne elementy układu asynchronicznego pracują w określonej kolejności Każdy okres, podczas którego w układ. Nie zachodzi żadna zmiana jest nazywany taktem.

Tabele prawdy bramek logicznych

NOT

IN	**OUT**
0	1
1	0

OR

IN1	IN2	OUT
0	0	0
1	0	1
0	1	1
1	1	1

AND

IN1	IN2	OUT
0	0	0
1	0	0
0	1	0
1	1	1

NAND

IN1	IN2	OUT
0	0	1
1	0	1
0	1	1
1	1	0

NOR

IN1	IN2	OUT
0	0	1
1	0	0
0	1	0
1	1	0

XOR

IN1	IN2	OUT
0	0	0
1	0	1
0	1	1
1	1	0

Multipleksery

13-03-2008

Jeśli mamy bardzo dużo źródeł danych to nie możemy ich jednocześnie „rzucić” na ekran, ponieważ wtedy stało by sie to nieczytelne dla człowieka. Dlatego trzeba jakoś podzielić sygnał (zaadresować by moc go później odczytać).

Multiplekser jest układem komutacyjnym (przełączającym), posiadającym ***k*** wejść informacyjnych (zwanych też wejściami danych), ***n*** wejść adresowych (sterujących) (zazwyczaj $k=2^n$) i jedno wyjście ***y***. Posiada też wejście sterujące działaniem układu oznaczane ***S*** (ang. *strobe*) lub ***e*** (ang. *enable*).

Jego działanie polega na połączeniu jednego z wejść x_i z wyjściem y. Numer wejścia jest określany przez podanie jego numeru na linie adresowe **A**. Jeśli na wejście strobujące (blokujące) ***S*** podane zostanie logiczne zero, to wyjście ***y*** przyjmuje określony stan logiczny (zazwyczaj zero), niezależny od stanu wejść **X** i **A.**

Sumator

27-03-2008

Podstawowymi elementami sumatora jest bramka XOR oraz bramka AND. Przy zsumowaniu dwóch cyfr otrzymujemy czasem przeniesienie(Carry).

Schemat sumatora 2 liczbowego(dalej nazwanego półsumatorem):

Sumator dla 4 cyfr:

s

Ogólna architektura mikroprocesora

03-04-2008

Mikrprocesor 8086 jest procesorem 16-bitowym wprowadzonym na rynek w 1980 roku. Konstruktorzy firmy Intel wprowadzili wiele nowych rozwiązań w zakresie architektury, które nie były stosowane w mikroprocesorach 8-bitowych. Można wymienić następujące rozwiązania poczynione w tym procesorze:

* rozszerzenie listy rozkazów
* rozszerzenie możliwości adresowania operandów
* wprowadzenie segmentacji obszaru pamięci
* mechanizmy przyśpieszenia pracy
* mechanizmy dla pracy wieloprocesorowej

W mikroprocesorach 8-bitowych stosowano jedynie 4 tryby adresowania :rejestrowy, natychmiastowy, pośredni i bezpośredni natomiast w 16 bitowych procesorach dodano jeszcze 3 tryby adresowania poprzez dodanie rejestr ow bazowych i rejestrów indeksowych :indeksowy, bazowy i indeksowo-bazowy. Dla rozdzielenia obszarów przeznaczonych dla programu danych i stosu wprowadzono mechanizm segmętacji. Mikroprocesor zawiera cztery rejestry segmentowe, w których przechowywane są adresy początków segmentów. Zawartości tych rejestrów wraz z adresem efektywnym, obliczanym przez procesor w zależności od trybu adresowania, stanowią

adresy fizyczne pamięci. Taki sposób adresowania ułatwia relokację programów i danych. W czasie wykonywania rozkazów mikroprocesor musi czekać na odpowiedź pamięci, np. na kod operacyjny rozkazu w cyklu pobierania rozkazu. W tym czasie mikroprocesor może być zajęty obliczaniem adresu efektywnego. Obie czynności angażują różne bloki mikroprocesora i mogą być wykonywane równocześnie. Aby tak było, mikroprocesor podzielono na dwie części, z których jedna jest odpowiedzialna za współpracę z otoczeniem, a druga za wykonywanie rozkazów. Ponadto zaprojektowano tzw. układ kolejki, który umożliwia wpisywanie i pamiętanie sześciu kolejnych bajtów pobieranych z pamięci w czasie, gdy mikroprocesor wykonuje inne czynności.

Mechanizmy wieloprocesorowości wprowadzone w mikroprocesorze 8086 można podzielić na dwie grupy: jedną dotyczącą współpracy z koprocesorem i drugą dotyczącą współpracy z innymi mikroprocesorami.

Mikroprocesor 8086 składa się z dwóch współpracujących jednostek,działających jednocześnie:

- Jednostki wykonawczej EU

- Zespołu łącza z magistralą systemową BIU

Jednostka Wykonawcza

zawiera blok arytmetyczno-logiczny ALU ,rejestr zanczników FR , blok rejestrów ogólnego przeznaczenia i układ stronowania. Blok ALU dołączony jest do magistrali wewnętrznej mikroprocesora. Z magistrali tej pobierane są argumenty operacji, a także wysyłany jest na nią wynik operacji.

Rejestr Znaczników rejest ten jest 16-bitowy ale wykorzystywane jest jedynie 9 bitów o następującym znaczeniu :

SF (*sign flag*) - znacznik znaku - równy najbardziej znaczącemu bitowi wyniku

ZF (*zero flag*) - znacznik zera

PF (*parity flag*) - znacznik parzystości - ustawiany w zależności od liczby jedynek w najniższych 8 bitach wyniku

AF (*auxiliary carry flag*) - znacznik przeniesienia połówkowego (pomocniczego)

CF (*carry flag*) - znacznik przeniesienia

OF (*overflow flag*) - znacznik nadmiaru

IF (*interrupt flag*) - znacznik przerwań

DF (*direction flag*) - znacznik kierunku, wskazuje, czy zawartości rejestrów SI i DI mają być zwiększane lub zmniejszane o jeden w czasie wykonywania operacji na ciągach

TF (*trap flag*) - znacznik pułapki umożliwiającej pracę krokową. Znacznik ten może być ustawiony za pomocą jedynki na odpowiedniej pozycji słowa stanu programu PSW (program status

Rejestry ogólnego przeznaczenia

- ***AX*** *lub* ***AH, AL*** *- rejestr akumulatora*
- ***BX*** lub ***BH***, ***BL*** - rejestr bazowy
- ***CX*** lub ***CH***, ***CL*** - rejestr zliczający
- ***DX*** lub ***DH***, ***DL*** - rejestr danych
- ***SP*** - wskaźnik stosu
- ***BP*** - wskaźnik bazy
- ***DI*** - rejestr adresu przeznaczenia
- ***SI*** - rejestr adresu źródłowego

Blok współpracy z magistralą systemową

Blok współpracy z magistralą systemową BIU zawiera blok rejestrów segmentowych i licznik rozkazów IP, sumator służący do obliczenia adresu fizycznego, układ kolejki rozkazów oraz blok sterowania.Jednocześnie pobieranie rozkazów z pamięci operacyjnej i ich wykonywanie jest możliwe dzięki zastosowaniu kolejki rozkazów. Układ ten pozwala na odczytywanie rozkazów z pamięci przez BIU i ich zapamiętywanie oraz na jednoczesne pobieranie zapamiętanych rozkazów przez EU i ich wykonywanie.

BIU realizuje mechanizm segmentacji w celu umożliwienia odseparowania przestrzeni adresowych dla programów, dla danych i dla stosu, oraz łatwej ich relokacji (zmiany położenia). W mikroprocesorze 8086 są rozróżnione cztery logiczne obszary pamięci. Każdemu z nich przypisany jest jeden rejestr segmentowy:

CS - rejestr adresu segmentu programu - wskazuje na początek bloku pamięci lub na segment kodu , w który rezyduje następny do wykonania rozkaz
DS - rejestr adresu segmentu danych - wskazuje początek segmentu danych zawierających argumenty
SS - rejestr adresy segmentu stosu - wskazuje początek bloku pamięci zwanego segmentem stosu
ES - rejestr adresu segmentu dodatkowego - wykorzystywany w zależności od aktualnych potrzeb
FS - rejestr adresu segmentu dodatkowego - wykorzystywany w zależności od aktualnych potrzeb
GS - rejestr adresu segmentu dodatkowego - wykorzystywany w zależności od aktualnych potrzeb

Długość każdego segmentu wynosi 64 kB. Rejestry segmentowe mają również długość szesnastu bitów. Wskazują one adres fizyczny początku segmentu w przestrzeni adresowej pamięci o pojemności 1MB.

Tryby adresowania

Trybem adresowania nazywamy sposób wyznaczania adresu operandu, którego to mianem określamy argumenty i wyniki operacji. W mikroprocesorze 8086 każdy z rejestrów ogólnego przeznaczenia może służyć do przechowywania adresu lub jego składnika.

Mikroprocesor 8086 realizuje następujące **tryby adresowania:**

Adresowanie natychmiastowe

W adresowaniu natychmiastowym argument pobierany jest bezpośrednio z rozkazu. Np. MOV AX, 0x20 – w rejestrze AX zostanie zapisana liczba 32.

Adresowanie rejestrowe

W adresowaniu rejestrowym operandy znajdują się w rejestrach wewnętrznych mikroprocesora. Np. MOV AX, BX – w rejestrze AX zostanie zapisana zawartość rejestru BX.

Adresowanie bezpośrednie

W adresowaniu bezpośrednim adres operandu znajduje się bezpośrednio w rozkazie. Np. MOV AX, [0xB800] – w rejestrze AX zostanie zapisana zawartość komórki pamięci (segment danych) o adresie 0xb8000 (pamięc karty graficznej VGA).

Adresowanie pośrednie

W trybie adresowania pośredniego odwołujemy się do jednego z rejestrów roboczych procesora

(np. BX) lub do komórki pamięci (np.19). W rejestrze (BX) zapisany jest numer komórki pamięci, do której trzeba sięgnąć aby odczytać tam zawarty adres i przenieść do drugiego rejestru (AX). Np. MOV AX, [CX] – w rejestrze AX zostanie zapisana zawartość komórki pamięci o adresie, który znajduje się w rejestrze CX.

Adresowanie bazowe

Adresowanie bazowe jest to rodzaj adresowania pośredniego, gdzie rozkaz wskazuje na jeden z rejestrów bazowych *BX* lub *BP* i może zawierać 8-; lub 16-bitową wartość stanowiącą lokalne przemieszczenie. Adresem efektywnym jest suma zawartości rejestru bazowego i przemieszczenia. Np. MOV AX, [BP].

Adresowanie indeksowe

Adresowanie indeksowe jest rodzajem adresowania pośredniego, gdzie adres efektywny jest sumą zawartości rejestru indeksowego *SI* lub *DI* i lokalnego przemieszczenia. Np. MOV AX, [SI].

Adresowanie bazowo-indeksowe

W adresowaniu bazowo-indeksowym, adres efektywny jest sumą zawartości jednego z rejestrów bazowych, jednego z rejestrów indeksowych i lokalnego przemieszczenia. Np. MOV AX, [SI+BP].

Rozkazy mikroprocesora 8086 można podzielić na następujące grupy:

- arytmetyczno-logiczne
- przesłań
- skoków, obsługi pętli, wywołań i powrotów z podprogramu
- dotyczące rejestrów segmentowych
- wykonujące operacje na ciągach słów
- wejścia/wyjścia
- inne

Procesory x86

Późniejsze wersje tego mikroprocesora (i386, i486, itd.) wprowadzają wiele zmian, jak na przykład 32 bitowe rejestry, tryb chroniony, stronnicowanie pamięci. W trybie chronionym wszystkie rejestry procesora są 32 bitowe, w przypadku rejestrów segmentowych mimo tego że są 32 bitowe, procesor i tak używa tylko dolnych 16 bitów. Dodatkowo został wprowadzony specjalny tryb „virutal 8086 mode” w którym procesor może „emulować” mikroprocesor 8086. Tryb chroniony dodatkowo udostępnia wiele możliwości zabezpieczenia jądra systemu operacyjnego przez aplikacjami użytkownika.

Assembler

Język programowania niskiego poziomu, w którym rozkazy kodu maszynowego zostały przedstawione w postaci skrótów mnemonicznych. Poszczególne polecenia są w trakcie kompilacji tłumaczone na postać zrozumiałą dla procesora komputera.(zwyczajowo asemblery) to rodzina języków programowania niskiego poziomu, których jedno polecenie odpowiada zasadniczo jednej instrukcji procesora. Języki te powstały na bazie języków maszynowych danego procesora poprzez zastąpienie kodów liczbowych instrukcji kodu maszynowego ich mnemonikami. Dzięki stosowaniu kilkuliterowych skrótów poleceń zrozumiałych dla człowieka pozwala to z jednej strony na tworzenie oprogramowania, z drugiej strony bezpośrednia odpowiedniość mnemoników oraz kodu maszynowy umożliwia zachowanie wysokiego stopnia kontroli programisty nad działaniem procesora. Składnia języka asemblera zależy od architektury procesora, ale i używanego asemblera, jednak zwykle autorzy asemblerów dla danego procesora trzymają się oznaczeń danych przez producenta.

Przykładowe instrukcje assemblera

mov dest, src

Kopiuje zawartość src do dest. Jeżeli dest=SS przerwania są wyłączane (za wyjątkiem wczesnych błędnych procesorów 808x). Niektóre procesory wyłączają przerwania, jeżeli dest jest którymkolwiek z rejestrów segmentowych.

int num

Wywołuje przerwanie programowe o wskazanym numerze, czyści TF i IF i wrzuca na stos CS i IP oraz ładuje CS:IP znalezione w tablicy wektorów przerwań. Wykonywanie programu zaczyna się od nowych wartości CS:IP

call procedura

Wrzuca IP (oraz CS przy długich wywołaniach) na stos i ładuje IP adresem procedury. Wykonywanie programu rozpoczyna się od adresu wskazywanego przez CS:IP.

jmp target

Bezwarunkowy skok do target. Skoki domyślnie są z przedziału od -32768 tdo32767 bajtów od instrukcji po instrukcji skoku. Skoki typu NEAR i SHORT powodują zmianę rejestru IP, a FAR zmianę rejestrów CS i IP.

push dane

Odkłada dane na stos aplikacji. Aktualny adres stosu zapisany jest w rejestrze sp

pop dane

Podnosi wartość ze stosu i zapisuje ją do „dane”. Aktualny adres stosu zapisany jest w rejestrze sp

Przykładowy program napisany w Assemblerze

Poniżej przedstawiam kod bardzo prostego programu w Asseblerze w składni intela przeznaczony dla kompilatora NASM działający pod systemami GNU/Linux.

```
; Hello world by Grzegorz Glinski
; Kompilacja:
; nasm -f elf -o hello.o hello.asm
; ld -o hello hello.o
; Uruchomienie:
; ./hello
[SECTION .text]
GLOBAL main
main: ; Glowna procedura apliacji
     mov eax, 4   ; eax: Numer funkcji systemowej 4 = sys_write()
     mov ebx, 1   ; ebx: Numer deskryptora pliku: 1 = stdout
     mov ecx, msg ; ecx: wskaznik na napis
     mov edx, (msg_end – msg) ; edx: ilosc bajtow do wpisania
     int 0x80 ; wywolanie funkcji systemowej
     mov eax, 1   ; eax:  Numer funkcji systemowej 1 = sys_exit()
     xor ebx, ebx ; ebx: kod wyjsca (xor wyzeruje rejestr ebx)
     int 0x80 ; wywolanie funkcji systemowej
     jmp $ ; Petla dla bezpieczenstwa

[SECTION .data]
msg: db 'Hello world from assembler!',10,13,0
msg_end:
```

Turbo debugger

15-05-2008

Turbo debugger jest oprogramowaniem pozwalającym na śledzenie uruchamianych programów .Pozwala on na uruchomienie fragmętu kodu i analizę zawartości dowolnych rejestrów ui obszarów pamięci. Program ten pozwala nam zarówno na wykonanie instrukcji krok po kroku jak i podgląd zdarzeń i określonym punkcie programu.program ten jest bardzo przydatny do analizy programów posiadających błędy spowodowane niezrozumieniem problemu , niewiedzy o działaniu danej instrukcji i po prostu pomyłek w zapisie.