Architektura procesora

Procesor 8086 - rejestry

Rejestry

- **AX (ang. Accumulator)** - jest wykorzystywany głównie do operacji arytmetycznych i logicznych.
- **BX (ang. Base Registers)** - rejestr bazowy, głównie wykorzystywany przy adresowaniu pamięci.
- **CX (ang. Counter Registers)** – rejestr często wykorzystywany jako licznik, np. przy instrukcji LOOP.
- **DX (ang. Data Register)** - rejestr danych, wykorzystywany przy operacjach mnożenia i dzielenia, a także do wysyłania i odbierania danych z portów.

Rejestry c.d.

- **SI (ang. Source Index)** - rejestr indeksujący pamięć, wskazuje obszar z którego przesyłane są dane. W połączeniu z DS tworzy adres logiczny DS:SI
- **DI (ang. Destination Index)** - rejestr indeksujący pamięć, wskazuje obszar, do którego przesyłane są dane. W połączeniu z ES, tworzy adres logiczny ES:DI
- **BP (ang. Base Pointer)** - rejestr stosowany do adresowania pamięci.
- **SP (ang. Stack Pointer)** - wskaźnik stosu.

Rejestry c.d.

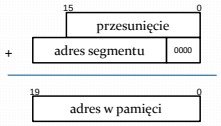
- **IP (ang. Instruction Pointer)** – zawiera adres aktualnie wykonywanej instrukcji, może być modyfikowany przez rozkazy sterujące pracą programu.
- **FLAGS** – rejestr znaczników.

Rejestry c.d. - segmentowe

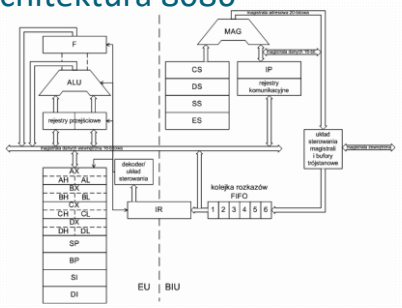
- **CS (ang. Code Segment)** - rejestr informujący o segmencie aktualnie wykonywanego rozkazu. Razem z IP tworzy adres logiczny CS:IP kolejnej instrukcji.
- **DS (ang. Data Segment)** - rejestr informujący o segmencie z danymi.
- **ES (ang. Extra Segment)** - rejestr informujący o segmencie dodatkowym np. przy operacjach przesyłania łańcuchów.
- **SS (ang. Stack Segment)** - rejestr informujący o segmencie stosu.

Adres w trybie rzeczywistym

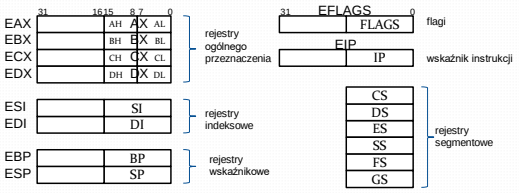
powstaje w wyniku sumowania położenia segmentu i przesunięcia w nim.



Architektura 8086



IA32- rejestry



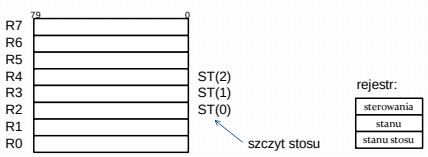
Rejestr flag

Rejestr flag w architekturze Intel x86			
bit	Skrót/wartość	opis	typ
0	CF	flaga przeniesienia (carry)	S
1	1	zarezerwowany	
2	PF	flaga parzystości (parity)	S
4	AF	flaga wyrównania (adjust)	S
6	ZF	flaga zera (zero)	S
7	SF	flaga znaku (sign)	S
8	TP	flaga umożliwiająca krokowe wykonanie (trap)	X
9	IF	flaga zezwolenia na przerwania (interrupt enable)	X
10	DF	flaga kierunku (direction)	C
11	OF	flaga przepełnienia (overflow)	S
12, 13	IOPL	poziom uprawnień we/wy (I/O privilege level, od 286)	X
14	NT	nested task flag (od 286)	X
16	RF	flaga wznowienia (resume, od 386)	X
17	VM	flaga trybu Virtual 8086 (od 386)	X
18	AC	alignment check (od 486SX)	X
19	VIF	Virtual interrupt flag (od Pentium)	X
20	VIP	Virtual interrupt pending (od Pentium)	X
21	ID	Identification (od Pentium)	X
3, 5, 15, 22–31	0	zarezerwowany	

S: Znacznik stanu
C: Znacznik kontrolny
X: Znacznik systemowy

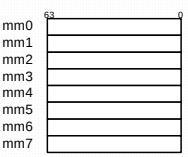
Koprocesor

stos rejestrów



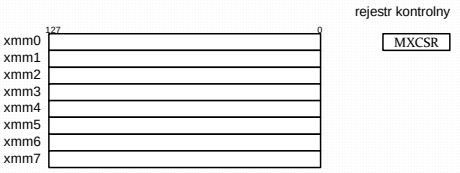
Rejestry MMX

Działają na nich instrukcje całkowitoliczbowe SIMD
Wykorzystują rejestry koprocesora

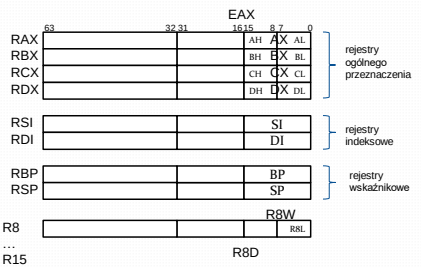


Rejestry XMM

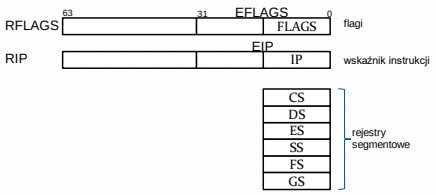
Działają na nich instrukcje zmiennoprzecinkowe SIMD



EM64T- rejestry

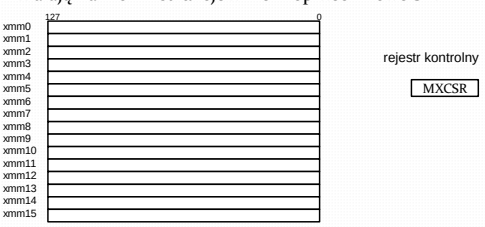


EM64T- rejestry



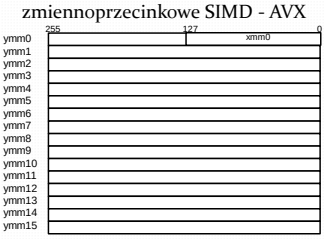
EM64T- rejestry XMM

Działają na nich instrukcje zmiennoprzecinkowe SIMD



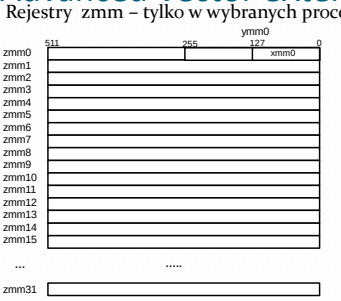
AVX-
Advanced Vector eXtensions

Rejestry ymm - działają na nich instrukcje zmiennoprzecinkowe SIMD - AVX

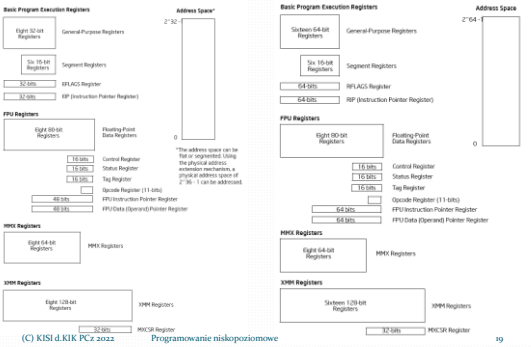


AVX512-
Advanced Vector eXtensions

Rejestry zmm – tylko w wybranych procesorach

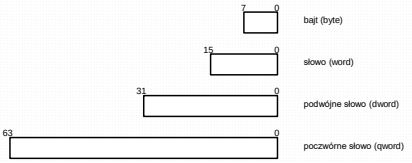


Środowisko 32 i 64 bitowe



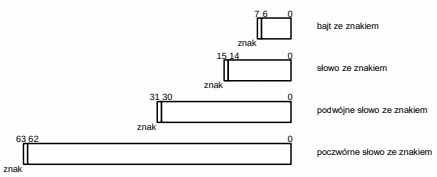
Liczbowe typy danych

Liczby całkowite bez znaku



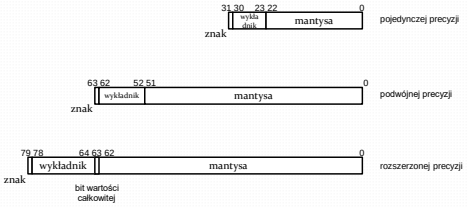
Liczbowe typy danych

Liczby całkowite ze znakiem

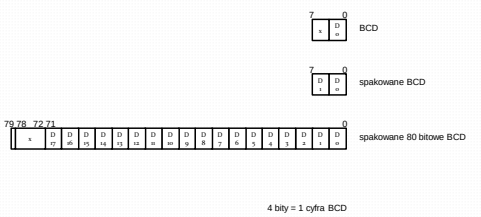


Liczbowe typy danych

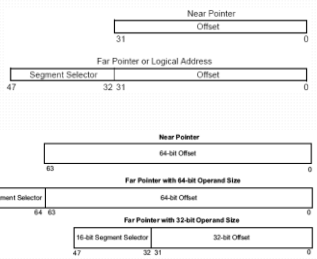
Liczby zmiennoprzecinkowe



Typy BCD

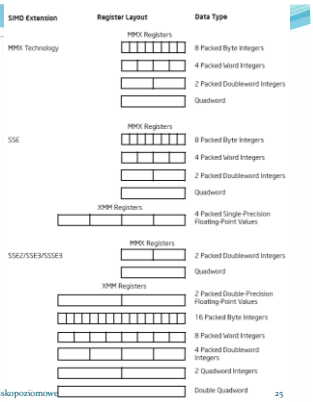


Wskaźniki w trybie 32 i 64 bitowym



SIMD

Rejestry i typy danych

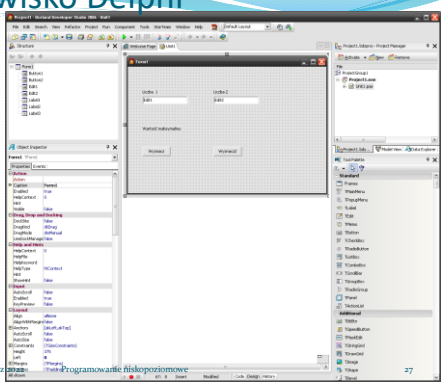


Kilka instrukcji

```
mov   eax,edx
mov   eax,[ebx]
mov   [rdx],rax
mov   eax,zmienna

push  bp
pop   eax

call  podprogram
ret
```

```
add   eax,edx
sub   rax,rbx
mul   ecx   ;edx:eax=eax*ecx
inc   ecx
dec   rcx

cmp   eax,ecx

jz    etykieta
jnz   etykieta
jc    etykieta
jnc   etykieta
```


Środowisko Delphi



Użycie rejestrów

Rejestry	Windows 32	Windows 64
do użycia	EAX, ECX, EDX, ST(0)-ST(7), K0-K7, xMM0-xMM7,	RAX, RCX, RDX, R8-R11, ST(0)-ST(7), K0-K7, xMM0-xMM5,xMM16-xMM31
do zabezpieczenia	EBX, ESI, EDI, EBP	RBX, RSI, RDI, RBP, R12-R15, xMM6-xMM15
parametry funkcji	cdecl, stdcall, pascal, Gnu C: na stosie, fastcall Microsoft/Gnu : ecx, edx fastcall Borland eax, edx, ecx thiscall Microsoft ecx	RCX, RDX, R8, R9 lub xMM0-xMM3 reszta na stosie.
zwracające wartość funkcji	EAX, EDX, ST(0)	RAX, xMM0

x = X, Y lub Z

Przykład

```
function TForm1.MyMax(x,y:integer):integer;
asm
  mov eax,x
  cmp eax,y
  jnc @@exit
  mov eax,y
@@exit:
//  mov result,eax
end;

function TForm1.MyMax2(x,y:integer):integer;
begin
  if x>y then
    MyMax2:=x
  else
    Result:=y;
end;
```


Kod wynikowy