Organizacja i architektura komputerów [1]

Wykład 7

Piotr Patronik

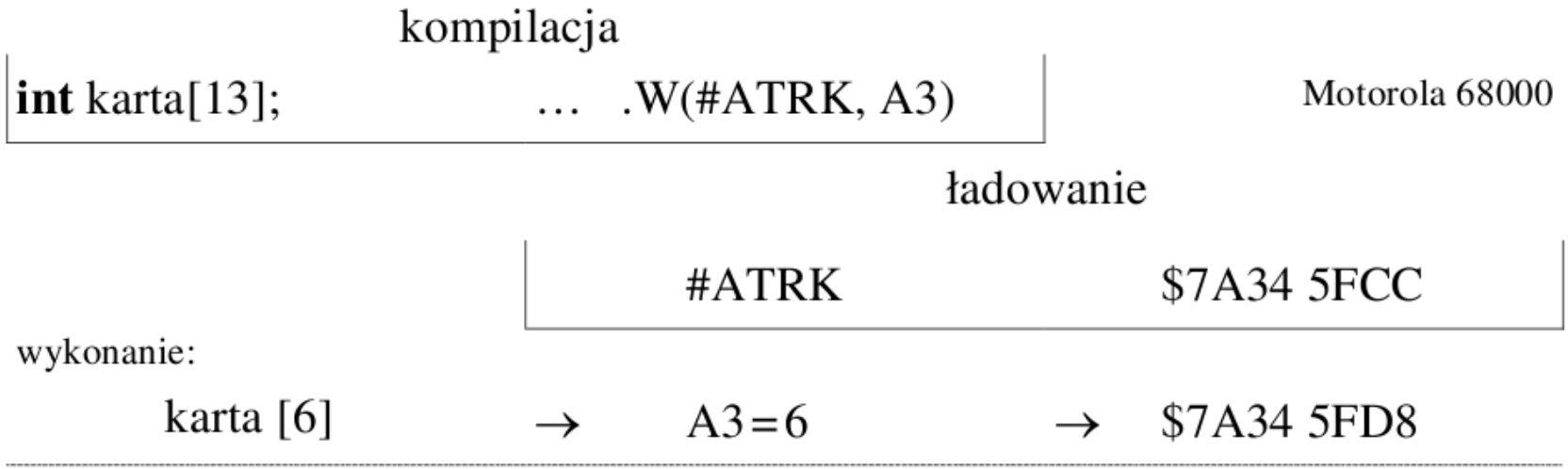
19 kwietnia 2016

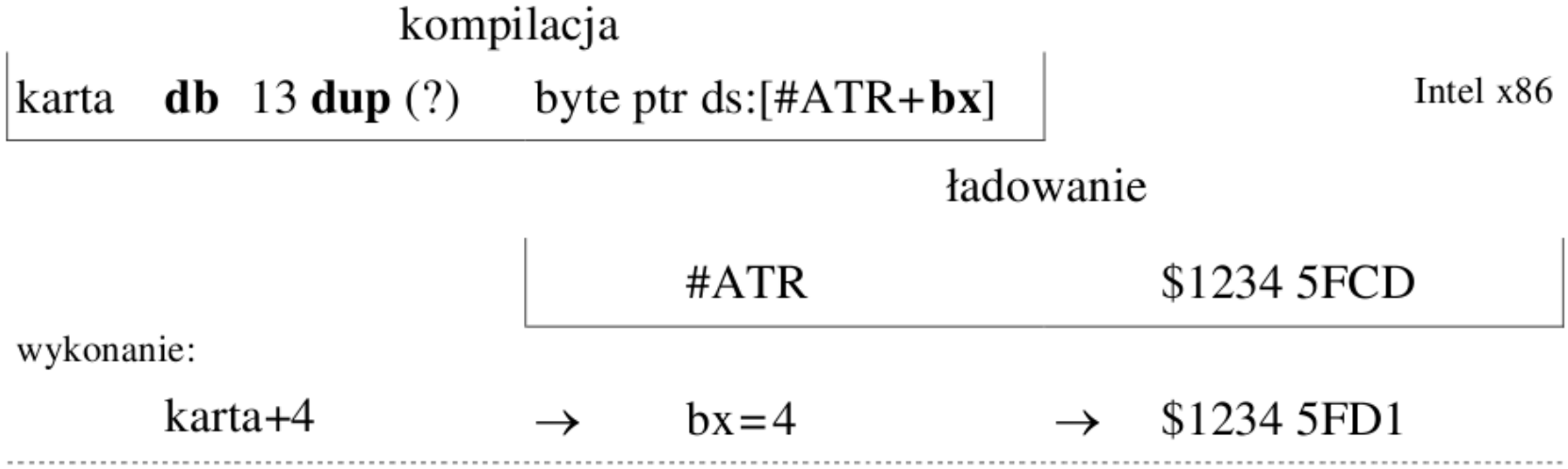
[1] (Prawie) dokładna kopia slajdów dr hab inż. J. Biernata

Nazwy i adresy

- *Nazwa* – identyfikator obiektu (zmiennej, etykiety) w języku programowania
  - *indeks* – identyfikator elementu obiektu (pojedynczej zmiennej)
- *Adres* – identyfikator (elementu) obiektu w języku maszynowym
- *Lokacja* – umiejscowienie obiektu w pamięci operacyjnej komputera
  Konieczne odwzorowanie (mapping) obiektów:
  nazwa (+indeks) $\longrightarrow$ *adres* $\longrightarrow$ *lokacja*
- *Adresowanie asocjacyjne (skojarzeniowe)*
  - dynamiczne, wzajemnie niezależne *powiązanie* lokacji z nazwą,
  - odwzorowanie nazwy w lokację (czasochłonne i skomplikowane) (pamięć podręczna)
- *Kompilacja* – *powiązanie* (binding) adresu z nazwą, przypisanie nazwie adresu
- *Ładowanie* – *powiązanie* lokacji z adresem, przypisanie adresowi lokacji w pamięci operacyjnej komputera

Powiązania


kompilacja
int karta[13];
… .W(#ATRK, A3)
Motorola 68000
ładowanie
#ATRK
$7A34 5FCC
wykonanie:
karta [6]
→
A3=6
→
$7A34 5FD8


kompilacja
karta db 13 dup (?)
byte ptr ds:[#ATR+bx]
Intel x86
ładowanie
#ATR
$1234 5FCD
wykonanie:
karta+4
→
bx=4
→
$1234 5FD1

Powiązania (2)

- Powiązania nazw z adresami – realizacja programowa (*software binding*)
- Powiązania adresów z lokacjami – realizacja sprzętowa (*hardware binding*)
- Powiązania programowe
  - zmiennych pojedynczych
  - zmiennych grupowych – *struktur danych* (*data structures*)
  - odwzorowują regularne obiekty, takie jak macierze, kolejki, stosy
    - nazwa obiektu $\longrightarrow$ adres obiektu
    - indeks elementu $\longrightarrow$ indeks adresowy wewnątrz obiektu
- Powiązania sprzętowe
  - wskaźniki adresowe $\longrightarrow$ adresy – *tryb adresowania* (addressing *mode*)
  - {adres obiektu, indeks elementu} $\longrightarrow$ adres w logicznej lub wirtualnej przestrzeni adresowej (adres efektywny)
  - adres efektywny $\longrightarrow$ lokacja
    - adres efektywny $\longrightarrow$ adres w rzeczywistej przestrzeni adresowej

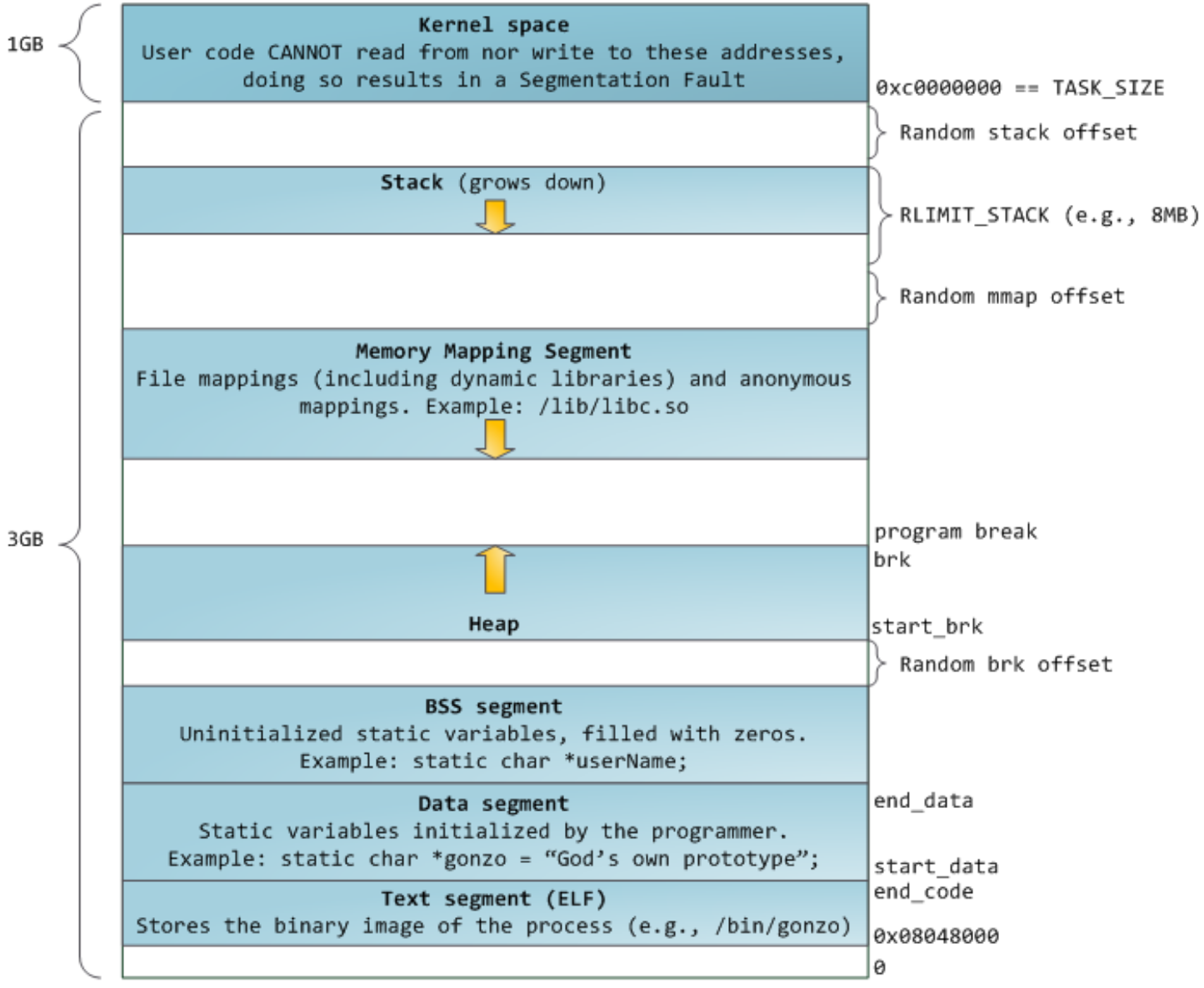
Przestrzeń adresowa

spójny zbiór lokalizacji o jednolitym trybie dostępu

- Każdy obiekt programowy jest umieszczony w przestrzeni adresowej
- Każdy obiekt programowy ma przypisaną lokalizację w przestrzeni adresowej
- Wskaźnik lokalizacji jest specyficzny (unikalny) dla każdej wyróżnionej przestrzeni adresowej
- Zakres wartości wskaźnika lokalizacji wyznacza rozmiar przestrzeni adresowej
- Przestrzeń adresowa może być abstrakcyjna
  – definiowana na poziomie języka programowania

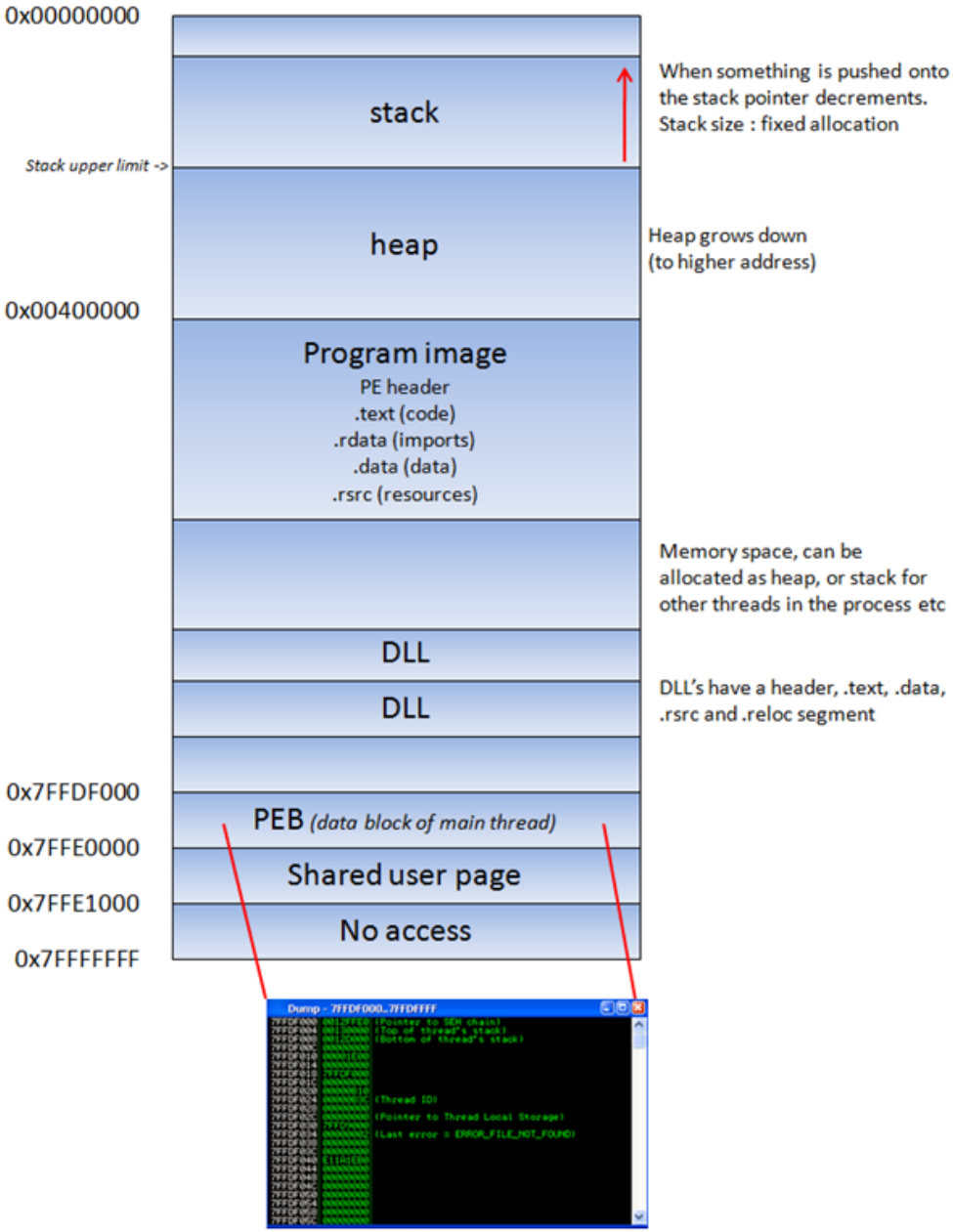
Przestrzeń adresowa – przykład

Mapa pamięci procesu (przybliżona) w systemie Linux



Przestrzeń adresowa – przykład (2)

Mapa pamięci procesu (przybliżona) w systemie Windows



Klasyfikacja przestrzeni adresowych

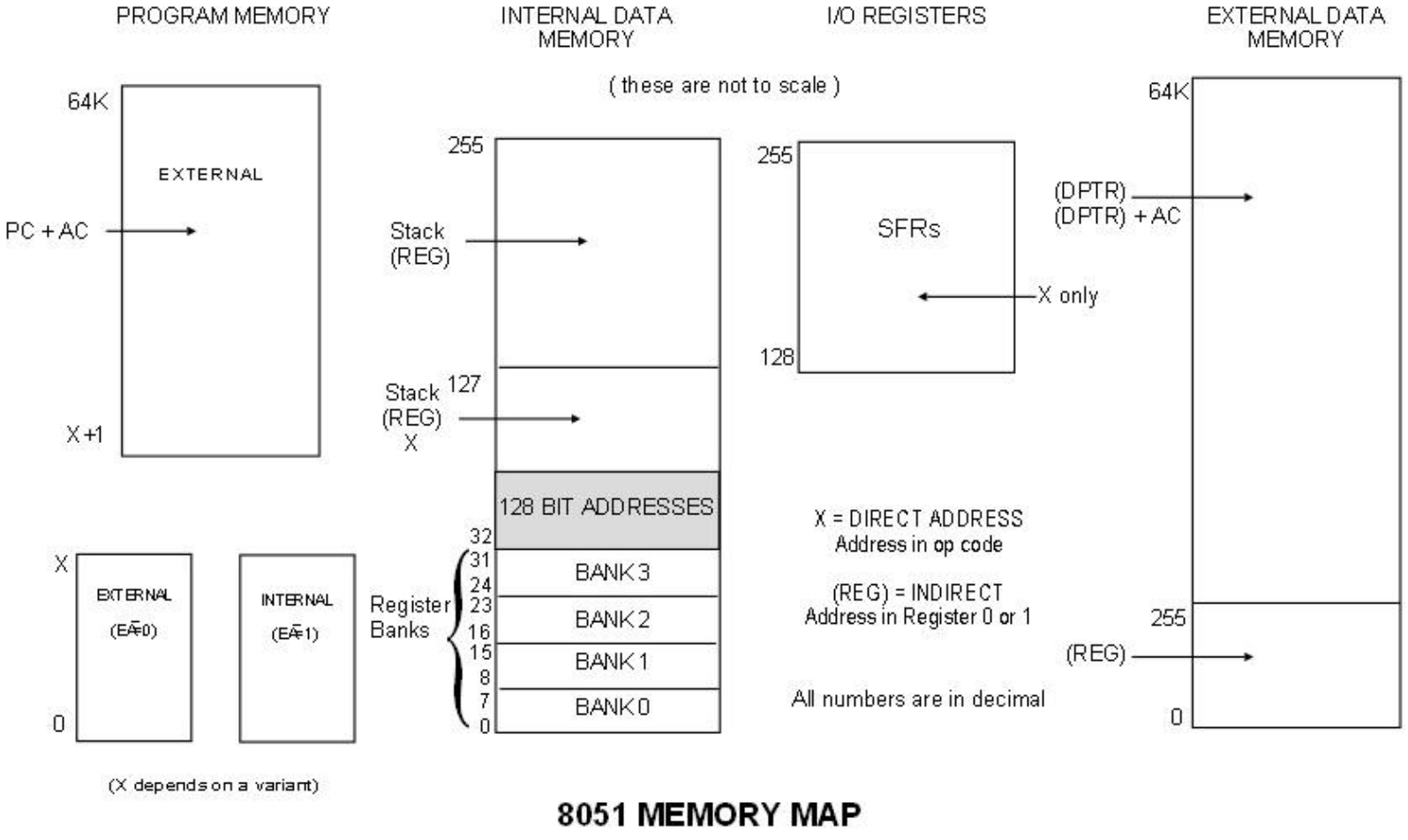
- przestrzeń rejestrów (*register space*)
  - obszar roboczy (*working store*) – rejestry procesora (*GPR*, *FPR*)
  - obszar sterowania (*control space*) – rejestry stanu (*SR*) i sterujące (*CR*)
- przestrzeń pamięci (*memory space*) – adresowanie swobodne (*random access*)
  - pamięć główna (*main memory space*)
  - przestrzeń peryferiów (*input/output, I/O space*)
  - stos (stack space)
  - przestrzeń wektorów przerwań (*interrupt vector space*)
- przestrzeń wirtualna
  - realizacja fizyczna bloków – pamięć zewnętrzna (dysk – sektory, ścieżki)
  - adresowanie opisowe, blokowe (tryb DMA)
    - parametry bloku – rozmiar, lokalizacja, opis transferu
    - lokalizacja obiektu wewnątrz bloku
- przestrzeń globalna – sieć Internet – adres opisowy

Separacja przestrzeni adresowych

- *fizyczna* – osobne układy dla różnych przestrzeni, rozróżnianie na podstawie rodzaju i/lub wartości wskaźnika lokacji
  - *architektura harwardzka*
    - *pamięć programu* | *pamięć danych* | *rejestry urządzeń*
  - *architektura klasyczna*
    - *komórki pamięci* | *rejestry urządzeń (peryferiów)*
- *logiczna* – rozdzielenie na poziomie architektury ISA (kodów rozkazów),
  osobne instrukcje dla różnych przestrzeni
  - *przestrzeń sterowania* – *ochrona sterowania* (rozkazy uprzywilejowane)
  - *przestrzeń peryferiów* – *rozdział umowny*
    - niezależna od separacji fizycznej
      Motorola 68K – brak rozkazów wejścia / wyjścia
    - opcjonalna – możliwe jednolite adresowanie
      Intel x86/Pentium – *specjalne rozkazy* (in, out) możliwe adresowanie jednolite (jak w pamięci)
    - zbędna w architekturze RISC – rozkazy load/store

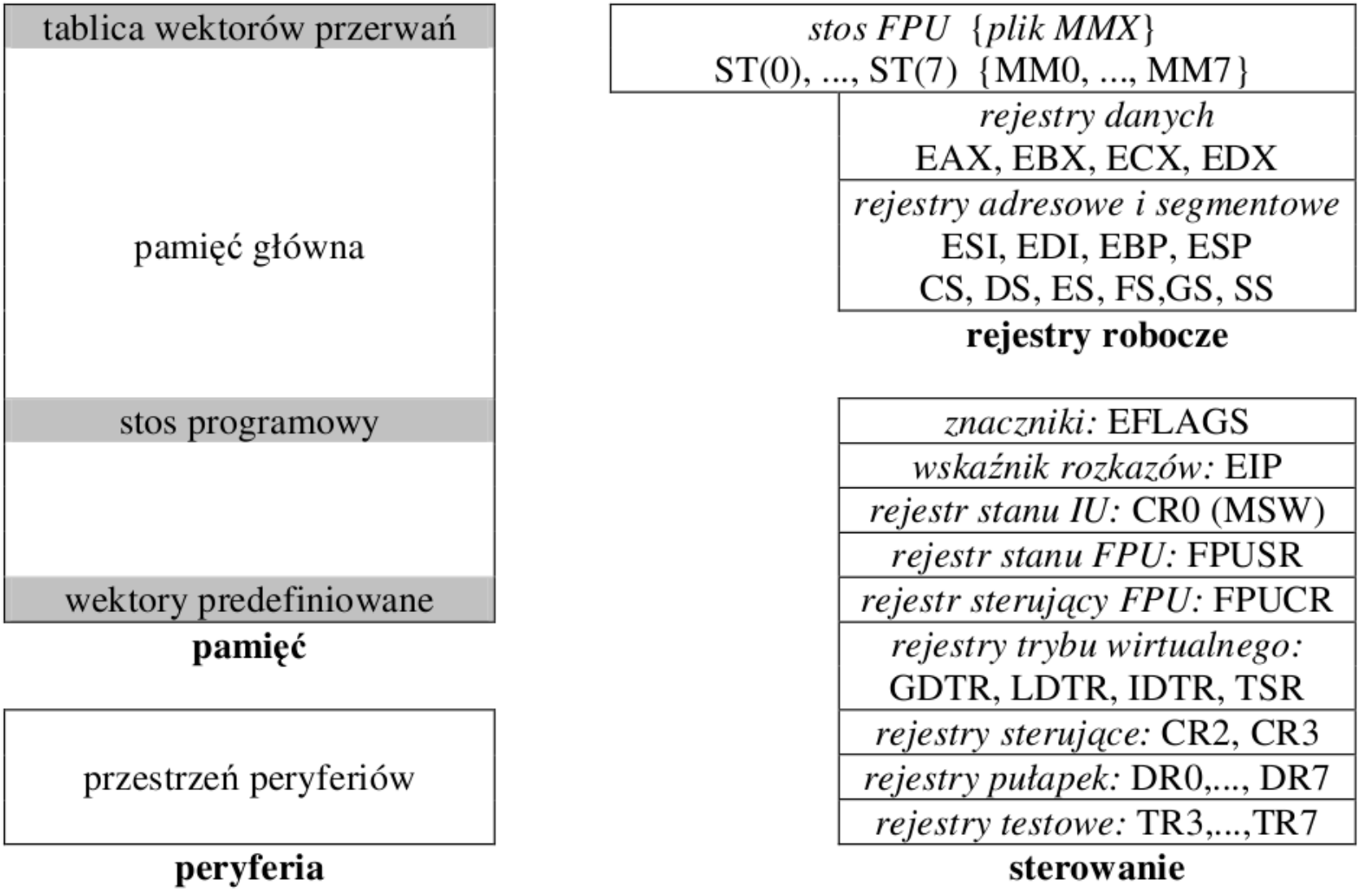
Przestrzeń adresowa procesora 8051

Memory Map.JPG

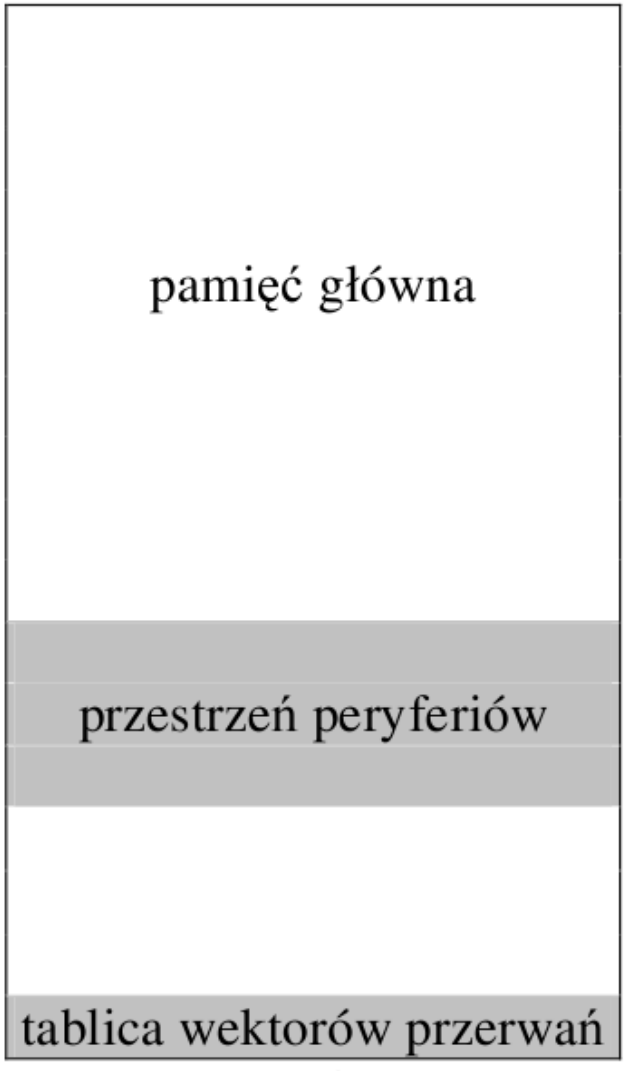


8051 MEMORY MAP

Przestrzenie adresowe procesorów Intel 80486 i Pentium II



Przestrzenie adresowe procesorów PowerPC 601 (Motorola)



pamięć

rejestry zmiennoprzecinkowe
FPR0–FPR31

rejestry stałoprzecinkowe
GPR0–GPR31

rejestry dedykowane
MQ, LR, CTR, RTC

rejestry użytkowe

rejestry segmentowe
SR0–SR15

rejestry systemowe

rejestr stanu: MSR

licznik rozkazów: PC

wyjątki: XER, FPXER

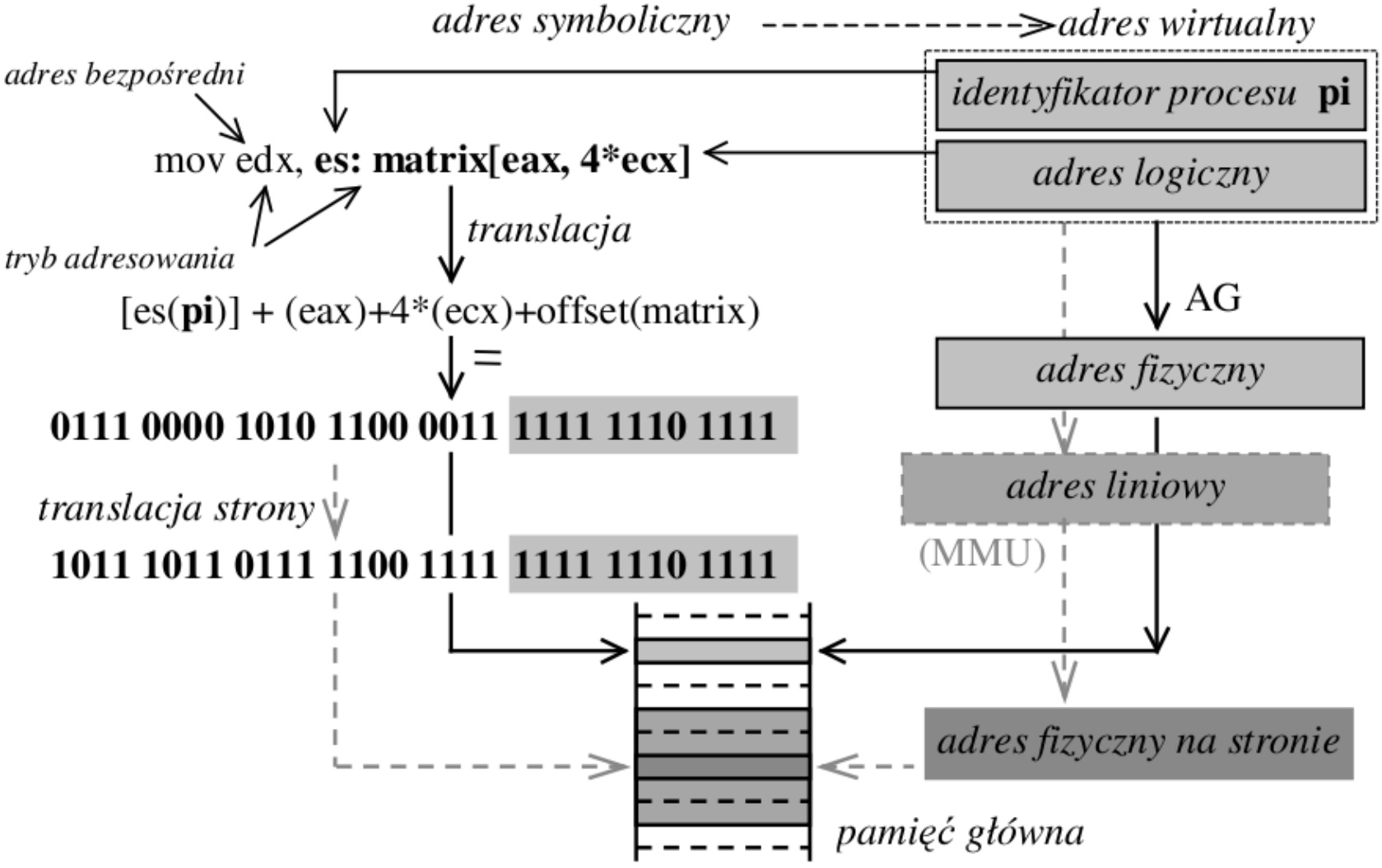
rejestr warunków: CR

rejestry specjalne

sterowanie

Tryb(y) adresowania

- sposób zamiany adresu symbolicznego na lokację w przestrzeni adresowej



Adresowanie bezpośrednie

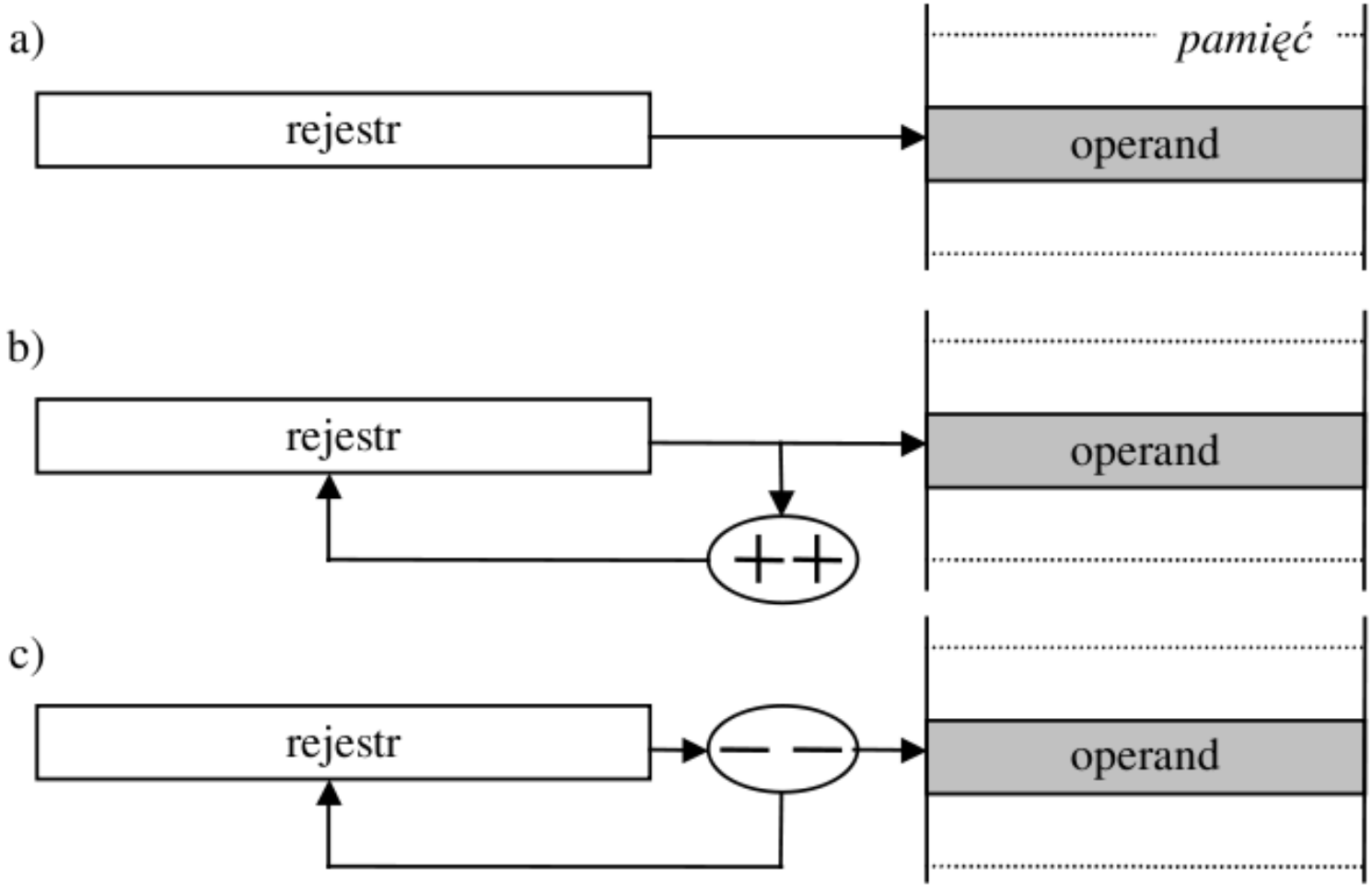
- adresowanie zeroelementowe
  kod rozkazu: | kod operacji | argument |
  - *błyskawiczne* (quick) – krótki kod danej w polu bitowym słowa kodu
    - adres argumentu – (licznik *rozkazów* : *pole kodu*)
  - *natychmiastowe* (immediate) – kod danej: rozszerzenie kodu rozkazu
    - adres argumentu – (licznik *rozkazów*++)
  - *zwarte* (compact) – operandy domniemane
- adresowanie jednoelementowe
  kod rozkazu: | kod operacji | adres argumentu |
  - *bezwzględne* (absolute) – adres danej w polu słowa kodu rozkazu
    - adres argumentu – (licznik *rozkazów*++)
  - *rejestrowe bezpośrednie* (register *direct*) – argument w rejestrze
    - adres argumentu – (licznik *rozkazów* : *pole kodu* – *numer rejestru*)

Adresowanie pośrednie

kod rozkazu:	kod operacji	$wsk\text{-}ad_1$	$wsk\text{-}ad_2$	...	przem.
kod rozkazu:	kod operacji	R_D	R_A	...	przem.

- adresowanie jednoelementowe
  - bezwzględne pośrednie (absolute indirect)
    – adres bezwzględny adresu danej jest słowem rozszerzenia kodu
  - rejestrowe pośrednie (register indirect) – adres danej w rejestrze
  - rejestrowe pośrednie z modyfikacją (register *indirect modified*)
    – adres automatycznie modyfikowany
    - zwiększany po użyciu (postinkrementacja)
    - zmniejszany przed użyciem (predekrementacja)
- adresowanie wieloelementowe
  – obliczanie wskaźnika na podstawie składowych

Adresowanie pośrednie jednoelementowe



a) pośrednie; b) z postinkrementacją; c) z predekrementacją

a)	mov eax, [ebx]		move d3,(a5)
b)	pop ecx	(*niejawne*)	cmpm (a5)+, (a7)+
c)	push eax	(*niejawne*)	abcd –(a3), –(a4)

Adresowanie pośrednie wieloelementowe

składowe adresu pośredniego

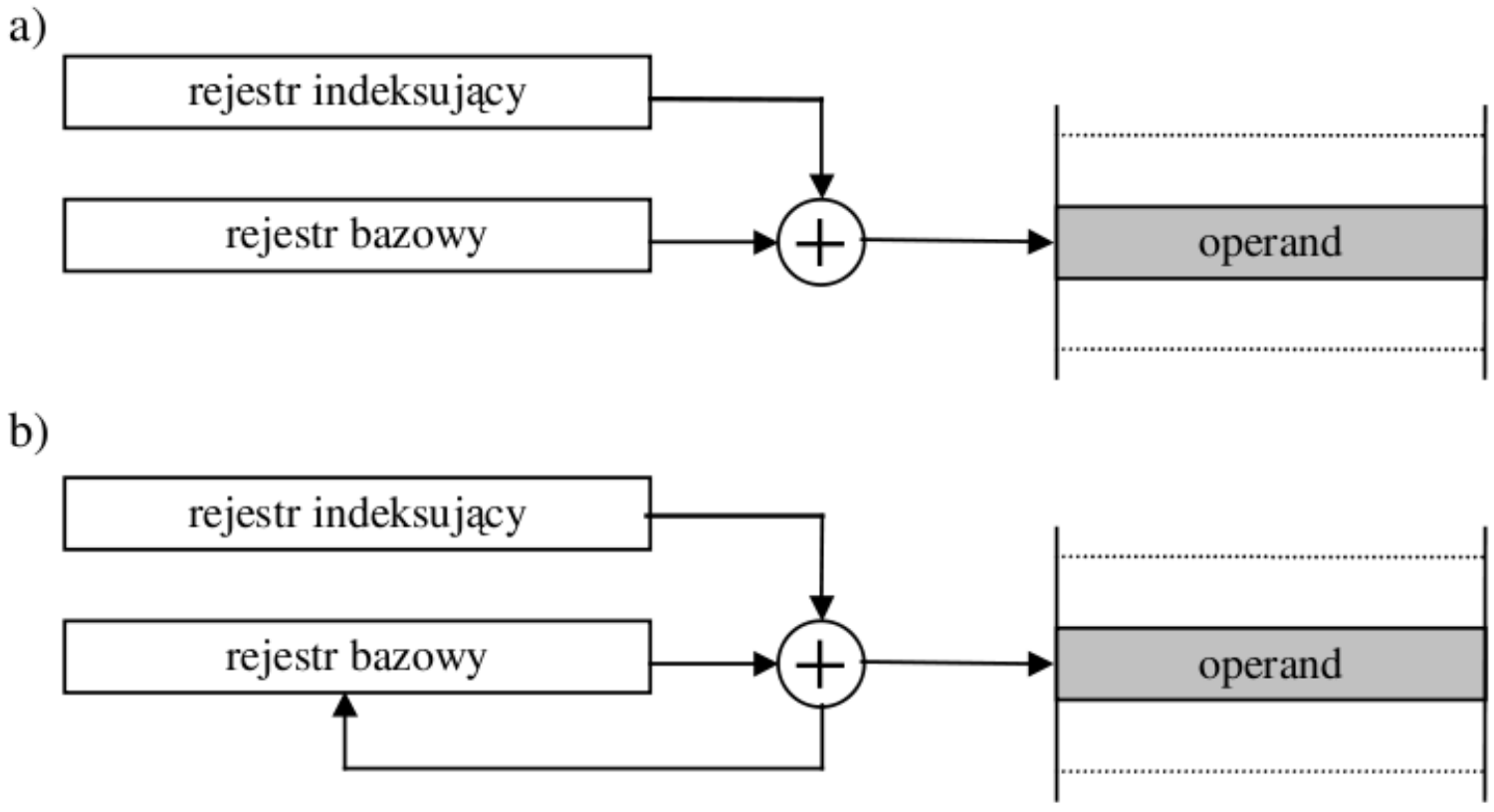
- baza (base) – wskaźnik adresu bloku (zawartość rejestru procesora)
- przemieszczenie bazy (offset) – stała, adres odniesienia (wskaźnika) bloku
  $\longrightarrow$ baza pośrednia – [baza + offset]
- indeks (index) – bieżący wskaźnik w bloku (zawartość rejestru procesora)
- skala (scale) indeksu – mnożnik indeksu wskazany w słowie kodu
- relokacja (outer displacement) – stała dodawana do obliczonego adresu

adresowanie dwuelementowe

- bazowe z przemieszczeniem (register indirect with offset)
- bazowe z przemieszczeniem i aktualizacją (register indirect updated)
- względne (PC-relative) z przemieszczeniem
- bazowo-indeksowe (base indexed)
- bazowo-indeksowe z aktualizacją (base-indexed updated)
- względne indeksowe (PC-based indexed)

adresowanie wieloelementowe jednopoziomowe (pośrednie)
adresowanie wieloelementowe dwupoziomowe (pośrednie)

Adresowanie bazowo–indeksowe z modyfikacją

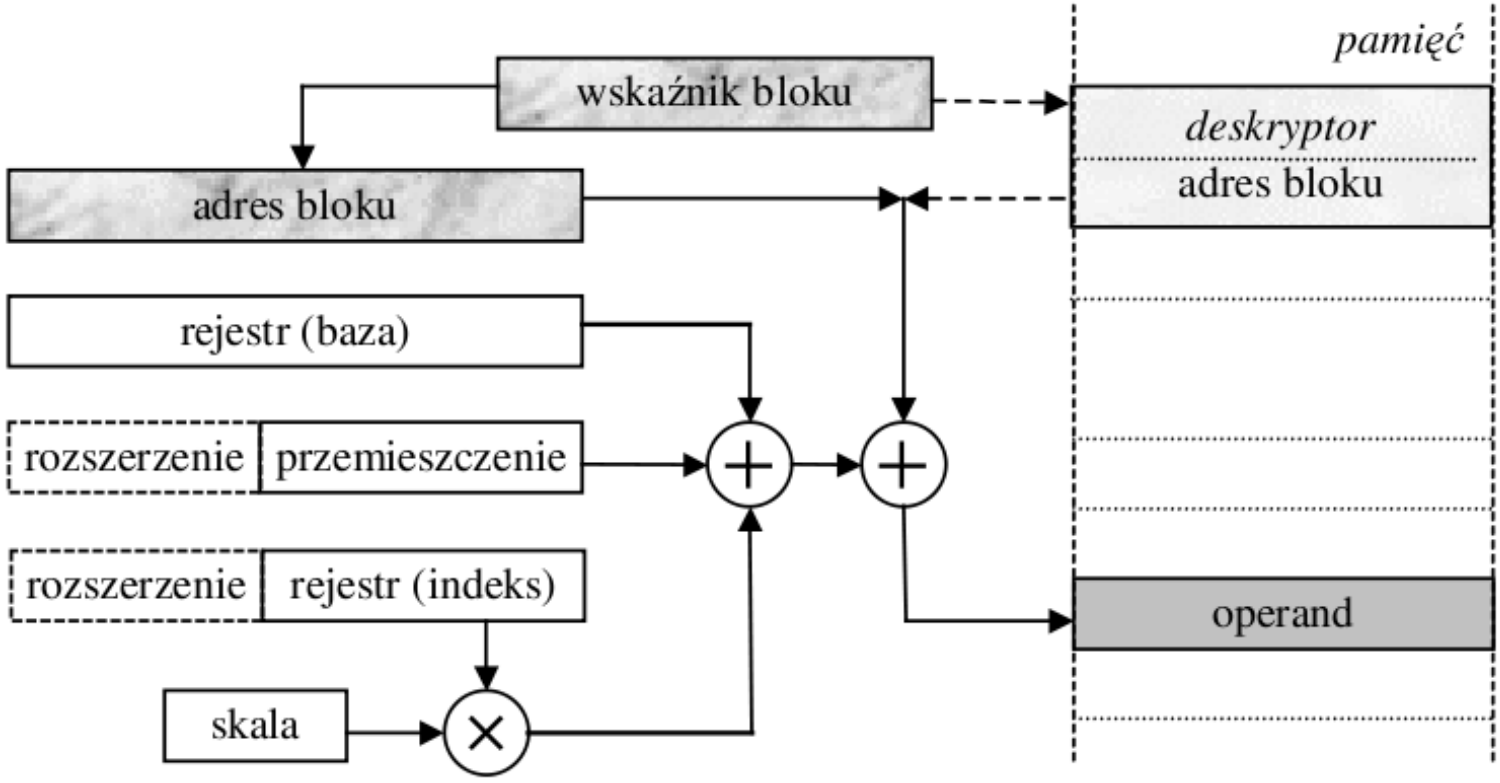


a) bazowo-indeksowe, b) bazowo-indeksowe z modyfikacją (PowerPC)

a)	lwz r1, r4, r7	mov eax, [bp, bx]
b)	lwzux r1, r4, r7	—

Adresowanie bazowo-indeksowe ze skalowaniem i deskryptorowe

LA=([wskaźnik segmentu])+(baza)+(indeks)×skala+przemieszczenie



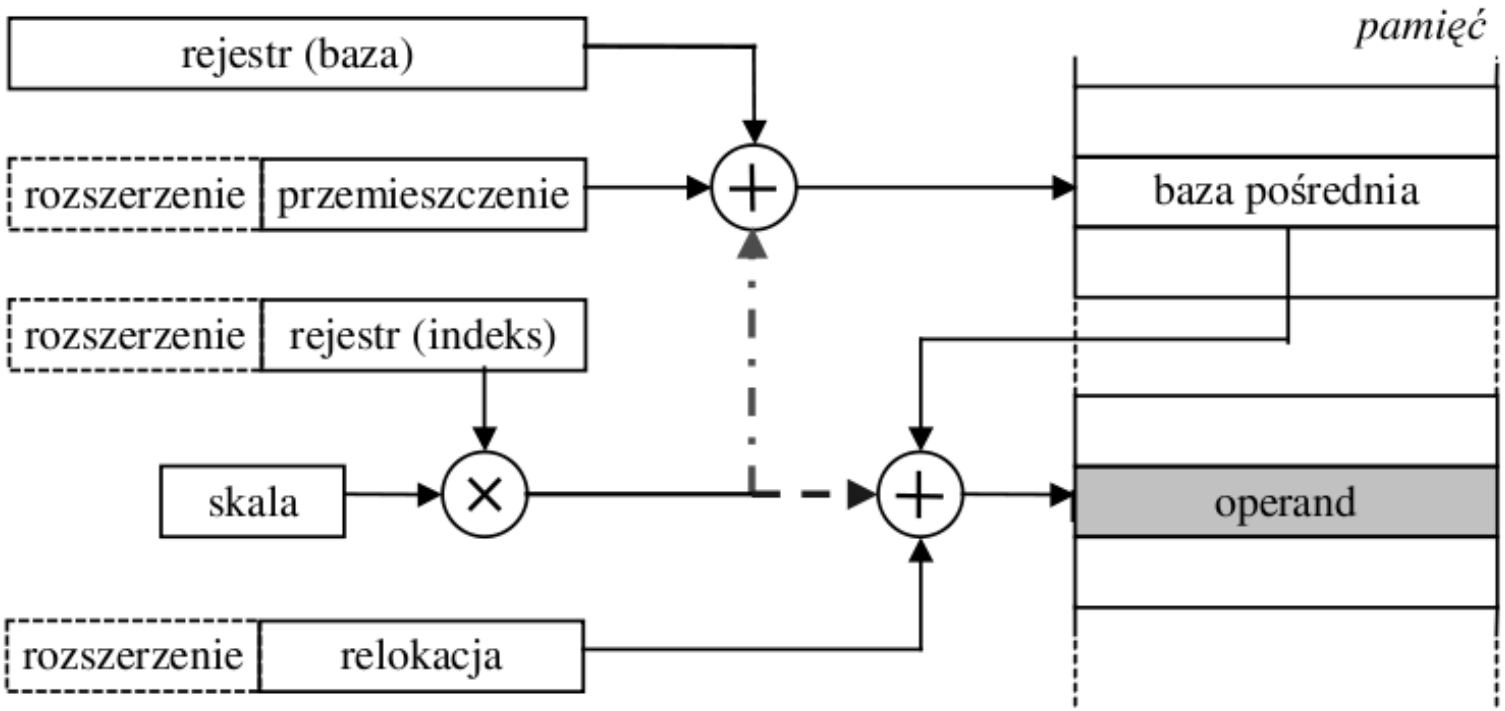
Adresowanie skalowane bazowo-indeksowe (i opisowe) (Intel 80386+)

```
add eax, matrix[ecx, 4*ebx] ;przem. [baza, sk*indeks]
```

Adresowanie preindeksowe i postindeksowe

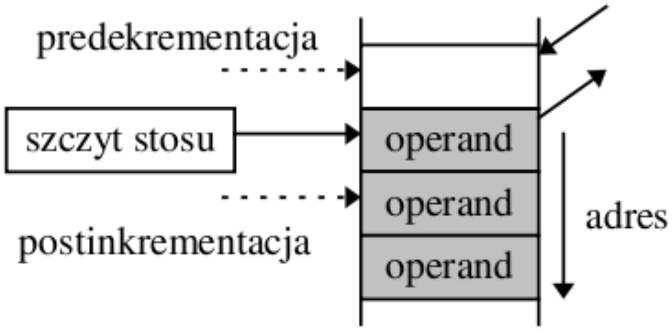
(pre) LA=[(baza)+przemieszczenie+(indeks)×skala]+relokacja

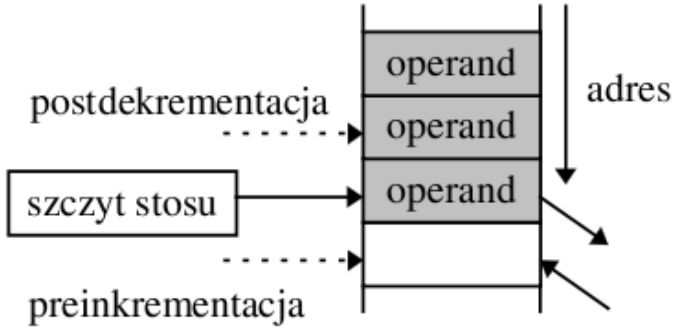
(post) LA=[(baza)+przemieszczenie]+(indeks)×skala+relokacja



- preindeksowe (– · – · –): `move d3, ([baza, a4, d4.w], relokacja)`
- postindeksowe (– – –): `move ([baza, a3], a2.w, relokacja), d5`

Adresowanie w obszarze stosu



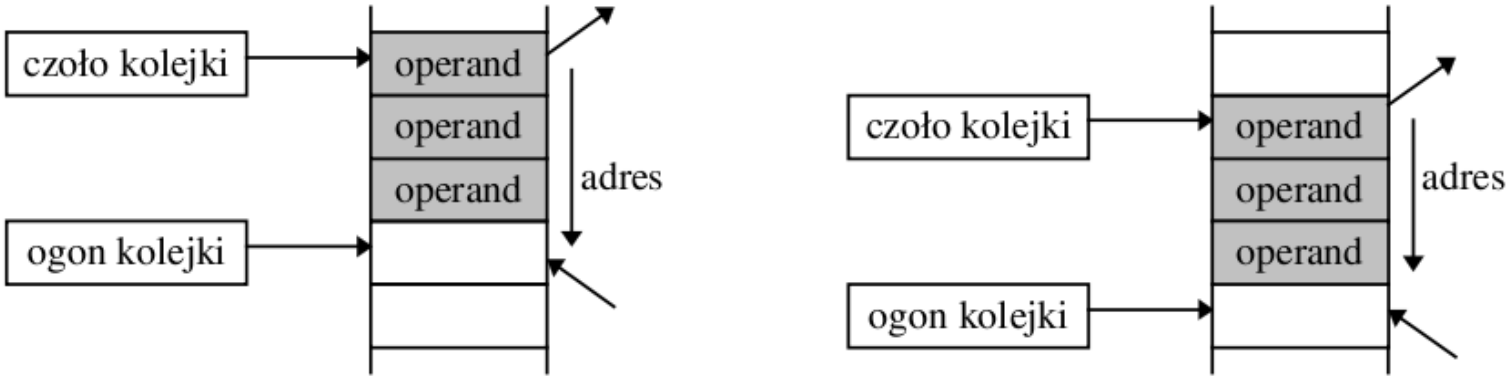


stos budowany w kierunku adresów malejących

stos budowany w kierunku adresów rosnących

- wskaźnik stosu (*stack pointer*, SP) – adres szczytu stosu
- adresowanie stosu – domniemane podczas wywołania procedur i funkcji
  zamierzone – specjalne rozkazy
- alternatywna definicja szczytu stosu jako lokacji najbliższej wolnej
  ! możliwa tylko wtedy, gdy struktura stosu jest jednorodna (każdy obiekt ma taki sam rozmiar)

Adresowanie kolejki



- kolejka sprzętowa – bufor sekwencji danych (na poziomie HSA)
- kolejka programowa
  - przemieszczanie spójnych bloków danych
  - programowanie współbieżne
  - system operacyjny – szeregowanie zadań

Tryby adresowania w architekturze CISC

- adresowanie stałych
  - natychmiastowe – powszechne (Intel x86/Pentium, Z80, . . .)
  - błyskawiczne – wyjątkowo i niejednolicie (Motorola 68K)
- adresowanie zwarte – częste, domniemany akumulator
- adresowanie rejestrowe bezpośrednie – ograniczone
- adresowanie rejestrowe pośrednie
  - jednoelementowe rzadko z automodyfikacją (Motorola 68K)
  - dwuelementowe
    - dwa rejestry, zwykle specjalizowane
    - rejestr + stała (pełny kod)
- adresowanie wieloelementowe
  - opis złożonych struktur danych
  - opis struktur pamięci programów współbieżnych

Tryby adresowania w architekturze RISC

- adresowanie stałych
  - błyskawiczne – typowe
  - stałe adresowe – różna interpretacja
  - natychmiastowe – wyjątkowo (pełne stałe adresowe – MIPS)
- adresowanie rejestrowe bezpośrednie – powszechne
  - wszystkie argumenty oprócz instrukcji load / store
- adresowanie rejestrowe pośrednie – argumenty instrukcji load i store
  - jednoelementowe często z autoindeksacją lub automodyfikacją
  - dwuelementowe
    - dwa dowolne rejestry
    - dowolny rejestr + stała (skrócony kod)
  - ze skalowaną autoindeksacją lub automodyfikacją
- adresowanie deskryptorowe
  - opis struktur pamięci programów współbieżnych

Tryby adresowania w architekturze Intel x86/Pentium i Motorola 68K

- adresowanie stałych
  ```
  add eax, 1245h   ; natychmiastowe (Intel)
  adq -3, d5       ; błyskawiczne (Motorola 68K)
  ```
- adresowanie zwarte
  ```
  imul ebx   ; domniemany eax (Intel)
  ```
- adresowanie rejestrowe bezpośrednie – ograniczone
  ```
  sub eax, ecx   ; (Intel)
  add d3, d5     ; (Motorola 68K)
  ```
- adresowanie rejestrowe pośrednie
  ```
  mov eax, [ebx, ebp]   ; rejestry specjalizowane (Intel)
  sub -(a3), -(a5)      ; z automodyfikacją (Motorola 68K)
  ```
- adresowanie wieloelementowe
  ```
  sub eax, blok[eax, 4*ecx]   ; rejestry specjalizowane (Intel)
  cmp (a3)+, d5               ; z automodyfikacją (Motorola 68K)
  ```

Tryby adresowania w architekturze Intel x86/Pentium (2)

Składnia AT&T – przykłady

1. Natychmiastowe: `mov $a, %eax`
2. Bezpośrednie: `mov a, %eax`
3. Pośrednie: `mov (%ebx), %eax`
4. Pośrednie z przemieszczeniem: `mov a(%ebx), %eax`
5. Pośrednie indeksowe: `mov (%ebx,%edi), %eax`
6. Pośrednie indeksowe z przesunięciem:
   `mov a(%ebx,%edi), %eax`
7. Pośrednie indeksowe z przemieszczeniem i skalowaniem:
   `mov a(%ebx,%edi,2), %eax`

- Rozkaz `lea` – załaduj adres efektywny

Tryby adresowania w architekturze RISC

- adresowanie stałych

```
adi.o r3, r5, -1   ; (PowerPC)
```

- adresowanie rejestrowe bezpośrednie – powszechne

```
add.o r3, r7, r15   ; (PowerPC)
```

- adresowanie rejestrowe pośrednie – argumenty instrukcji *load* i *store*

```
stw r5, r17            ; (PowerPC)
lwzux r5, r17, r18     ; z aktualizacją (PowerPC)
lwz r5, blok(r17)      ; (PowerPC)
```

- adresowanie deskryptorowe

```
mtsr sr7, r3   ; ładowanie r3 do rejestru segmentu sr7
mtsr r3, r7    ; ładowanie r3 do rejestru segmentu
               ; wskazanego przez 4 wyższe bity r7
```

Tryby adresowania w architekturze RISC (2)

Adresowanie ARM

1. Tryby błyskawiczny i rejestrowy
   - `mov r0, #4` (ograniczenia stałej)
   - `mov r0, r1`
2. Pośredni
   - `ldr r0, [r1, #4]`
   - `ldr r0, [r1, r2]`
   - `ldr r0, [r1, r2, lsl #4]`
3. Pośredni z preindeksacją
   - `ldr r0, [r1, #4]!`
   - `ldr r0, [r1, r2]!`
   - `ldr r0, [r1, r2, lsl #4]!`
4. Pośredni z postindeksacją
   - `ldr r0, [r1], #4`
   - `ldr r0, [r1], r2`
   - `ldr r0, [r1], r2, lsl #4`

- `lsl`, `lsr`, `asr`, `ror`, `rorx`

Adresowanie łańcuchów

adres		(ASCII)	
...000	0100 0101	64	*'d'*
...001	0110 0011	72	*'r'*
...010	0111 1000	61	*'a'*
...011	0001 0010	62	*'b'*
...			*„drab"*

ważniejszy niższy BE
(wysokokońcówkowy)

		(ASCII)	
	0001 0010	62	*'b'*
	0111 1000	61	*'a'*
	0110 0011	72	*'r'*
	0100 0101	64	*'d'*
			„drab"

ważniejszy wyższy LE
(wysokokońcówkowy)

adres łańcucha / słowa (offset – adres w bloku, względny)

Motorola 68K BE		Intel x86/Pentium LE	
notacja	*adres*	*notacja*	*adres*
tekst dc „drab"	adres tekst = = adres 'd'	tekst dw „drab"	offset tekst = = offset 'b'

Adresowanie liczb

- Program (gcc):

```
.data
d: .byte 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8

.text
.global main
main:
    nop
    int $3
    ret
```

- Wykonanie:

```
(gdb) x/8bx &d
0x601038: 0x00  0x01  0x02  0x03  0x04  0x05  0x06  0x07
(gdb) x/4hx &d
0x601038: 0x0100  0x0302  0x0504  0x0706
(gdb) x/2wx &d
0x601038: 0x03020100  0x07060504
(gdb) x/1gx &d
0x601038: 0x0706050403020100
(gdb)
```

Adresowanie etykiet (GDB)

- Program (gcc):

```
.data
d: .byte 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8
.text
e: .byte 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8
.global main
main:
    nop
    int $3
    ret
```

- Wykonanie:

```
(gdb) x/8bx d
0x3020100:    Cannot access memory at address 0x3020100
(gdb) x/8bx &d
0x601038:     0x00   0x01   0x02   0x03   0x04   0x05   0x06   0x07
```

- Ale...(sekcja `.text`/`.data`)

```
(gdb) x/8bx e
0x4004ed <e>:   0x00   0x01   0x02   0x03   0x04   0x05   0x06   0x07
(gdb) x/8bx &e
0x4004ed <e>:   0x00   0x01   0x02   0x03   0x04   0x05   0x06   0x07
(gdb)
```

Czasowe aspekty adresowania – wpasowanie słów

Rozdzielczość adresowania – fizycznie adresowalna jednostka informacji

- bit – szczególne przypadki (CISC – specjalne rozkazy dostępu)
- bajt – jednostka standardowa w pamięci fizycznej (organizacja bajtowa)
- słowo – jednostka logicznej organizacji programu (rozkazów i danych)

wpasowanie – umieszczenie słowa pod adresem podzielnym przez jego rozmiar