Program

PROGRAM = ALGORYTM + STRUKTURA DANYCH

algorytm – opis procedury przetwarzania danych

- **działania**:
  - arytmetyczne, logiczne
  - zmiany kodu, zmiany formatu
  - kopiowanie
- **dane**:
  - łańcuch bitów
  - umowy : kody znaków (ASCII), kody liczb (NB, U2, IEEE 754)
  - interpretacje – dane specyficzne (np. kolory, dźwięki, etc.)

struktura danych – wzajemne relacje zmiennych i ich atrybuty:

- identyfikator (wskaźnik, adres)
  – odwzorowanie w logicznej przestrzeni adresowej
- typ (predefiniowany lub strukturalny)
  – rozmiar jednostki (elementu zmiennej)
- rozmiar – całkowita liczba słów/bajtów

Obraz programu w pamięci

Proces: program w czasie wykonania

Ochrona informacji (procesu):

- partycje
- bloki (segmenty, sekcje) programu

Adresowanie informacji:

- w przestrzeni logicznej (programu)
  - względne → względem początku bloku (wskaźnik lokacji)
  - bezwzględne → ustalone części pamięci operacyjnej
- w przestrzeni fizycznej (procesu):
  - dynamiczne (zgodnie z przebiegiem przetwarzania)
    - blok algorytmu (kodu programu) → wskaźnik rozkazu (ang. *instruction pointer, program counter*)
    - blok stosu programowego → wskaźnik stosu (ang. *stack pointer*)
  - statyczne (zgodnie z opisem zmiennych)
    - blok danych (zmiennych), bufory danych → zgodnie z trybem adresowania (ang. *addressing mode*)

Nazwy i adresy

Nazwa – identyfikator obiektu (zmiennej, etykiety) w języku programowa
indeks – identyfikator elementu obiektu (pojedynczej zmiennej)

Adres – identyfikator (elementu) obiektu w języku maszynowym

Lokacja – umiejscowienie obiektu w pamięci operacyjnej komputera

Konieczne odwzorowanie (*mapping*) obiektów:

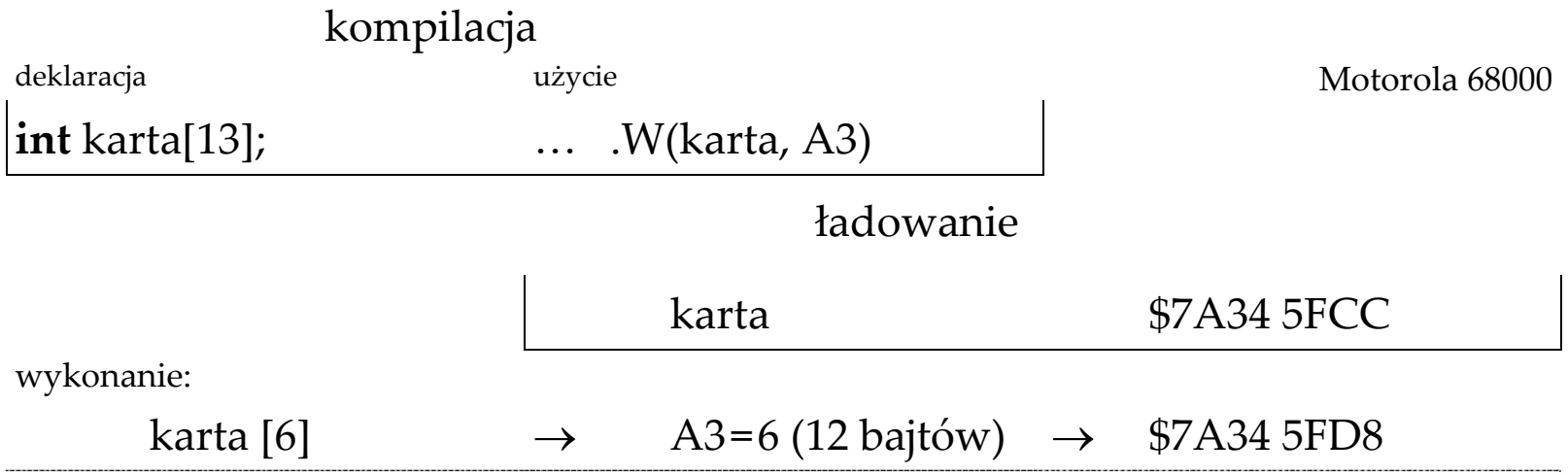
nazwa (+*indeks*) → *adres* → *lokacja*

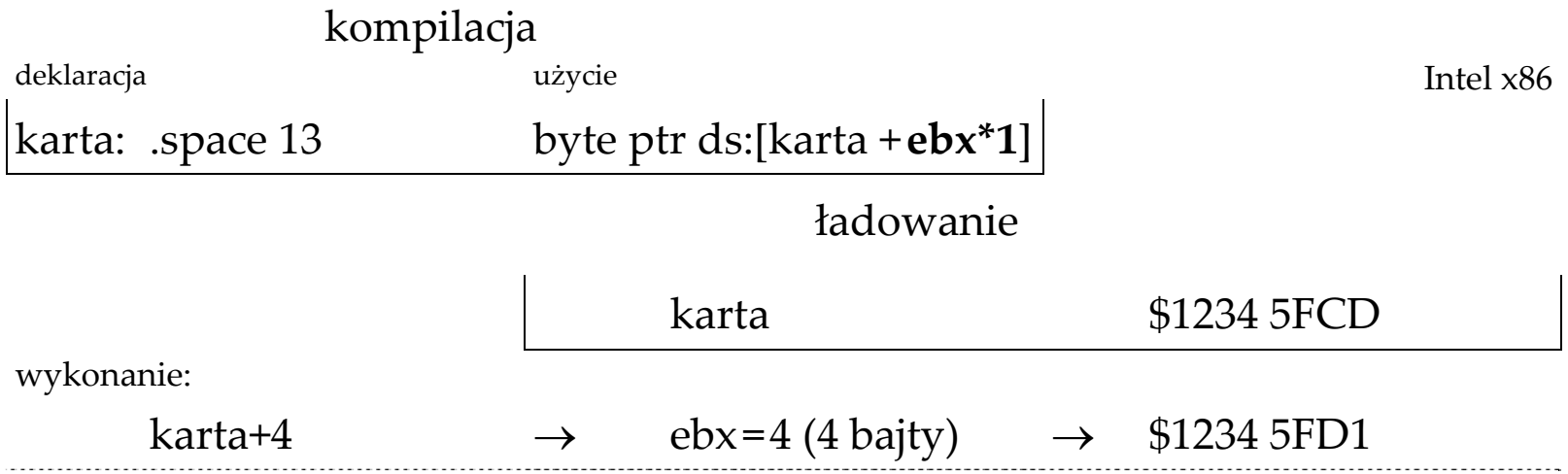
Adresowanie asocjacyjne (skojarzeniowe)
– dynamiczne, wzajemnie niezależne *powiązanie* lokacji z nazwą,
– odwzorowanie nazwy w lokację (czasochłonne i skomplikowane)

Kompilacja – *powiązanie* (*binding*) nazwy z adresem logicznym

Ładowanie – *powiązanie* adresu logicznego z adresem fizycznym, przypisanie adresowi logicznemu lokacji w pamięci operacyjnej komputera

Powiązania


kompilacja
deklaracja
użycie
Motorola 68000
int karta[13];
… .W(karta, A3)
ładowanie
karta
$7A34 5FCC
wykonanie:
karta [6]
→
A3=6 (12 bajtów)
→
$7A34 5FD8


kompilacja
deklaracja
użycie
Intel x86
karta: .space 13
byte ptr ds:[karta +ebx*1]
ładowanie
karta
$1234 5FCD
wykonanie:
karta+4
→
ebx=4 (4 bajty)
→
$1234 5FD1

Powiązania

Powiązania nazwy z adresem – realizacja programowa (ang. *software binding*)

Powiązania adresu fizycznego z lokacją – realizacja sprzętowa (ang. *hardware binding*)

Powiązania programowe

- zmiennych pojedynczych
- zmiennych grupowych – *struktur danych* (ang. *data structures*)
  – odwzorowują regularne obiekty, takie jak macierze, kolejki, stosy
  - nazwa obiektu → adres obiektu
  - indeks elementu → indeks adresowy wewnątrz obiektu

Powiązania sprzętowe

- wskaźniki adresowe → adresy – *tryb adresowania* (ang. *addressing mode*)
  - {adres obiektu, indeks elementu} → adres w logicznej lub wirtualnej przestrzeni adresowej (adres efektywny)
- adres efektywny → lokacja
  - adres efektywny → adres w rzeczywistej przestrzeni adresowej

Przestrzeń adresowa

*spójny zbiór **dozwolonych lokalizacji operandów** o jednolitym trybie dostępu*

Przestrzeń adresowa może być abstrakcyjna
– definiowana na poziomie języka programowania

- Każdy obiekt programowy ma przypisaną lokalizację w przestrzeni adresowej
- Wskaźnik lokalizacji (adres) jest specyficzny dla przestrzeni adresowej
- Rozmiar przestrzeni adresowej – zakres dozwolonych wartości adresów

Adresowanie bezpośrednie – kod operacji zawiera wartość lub adres argumentu

Przykłady: movd $5, %ead, movb %bl, %al
z = 5

Adresowanie pośrednie – kod operacji zawiera sposób obliczenia adresu operandu

Przykłady: addd %ead, tab(%ecx, %esi, 4), push %eax,
z = tab(i, j);

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*)

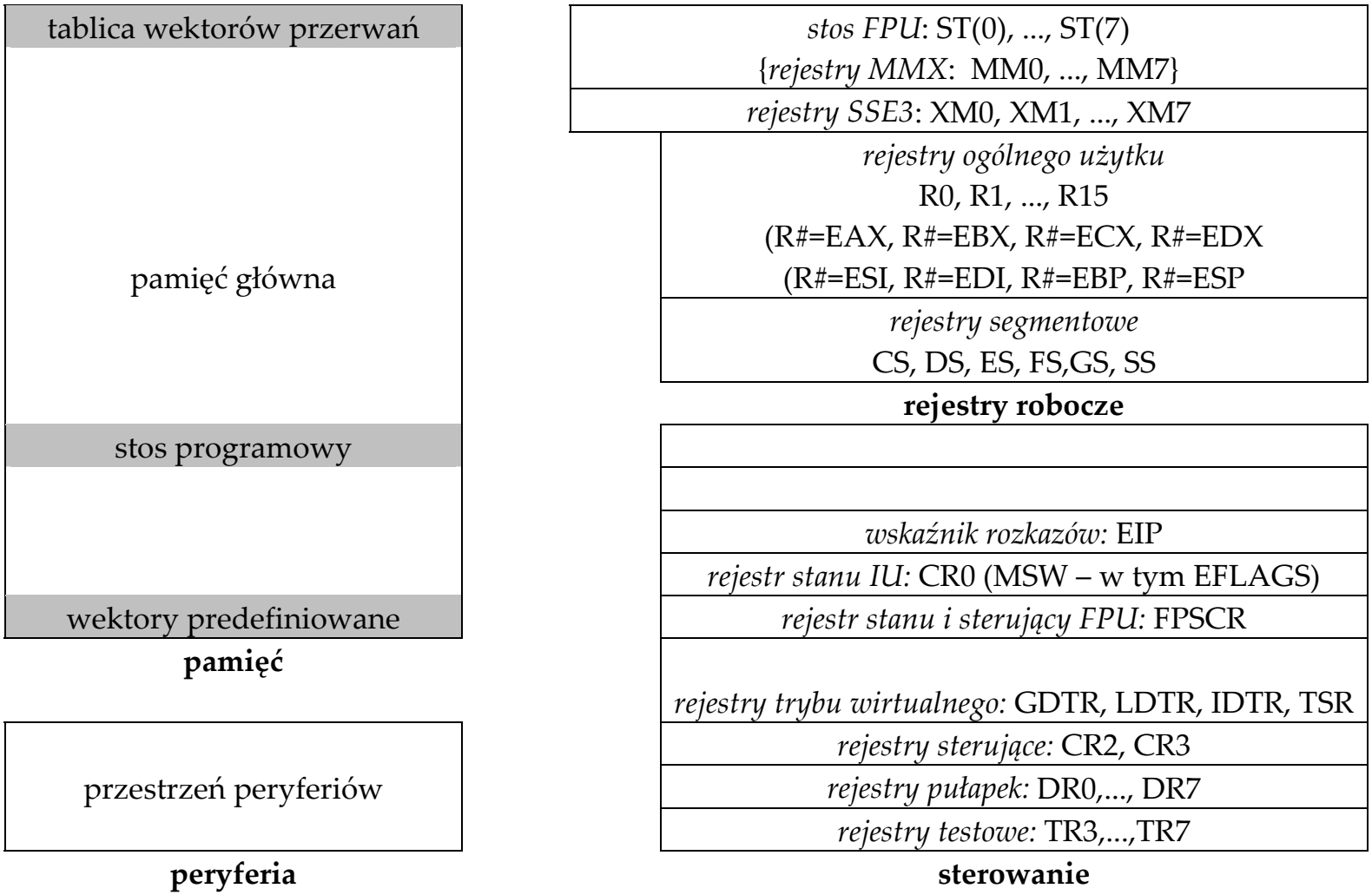
- przestrzeń wirtualna
  - realizacja fizyczna bloków – pamięć zewnętrzna (dysk – sektory, ścieżki)
  - adresowanie opisowe, blokowe (tryb DMA)
    - → parametry bloku – rozmiar, lokalizacja, opis transferu
    - → lokalizacja obiektu wewnątrz bloku

- przestrzeń globalna – sieć Internet – adres opisowy

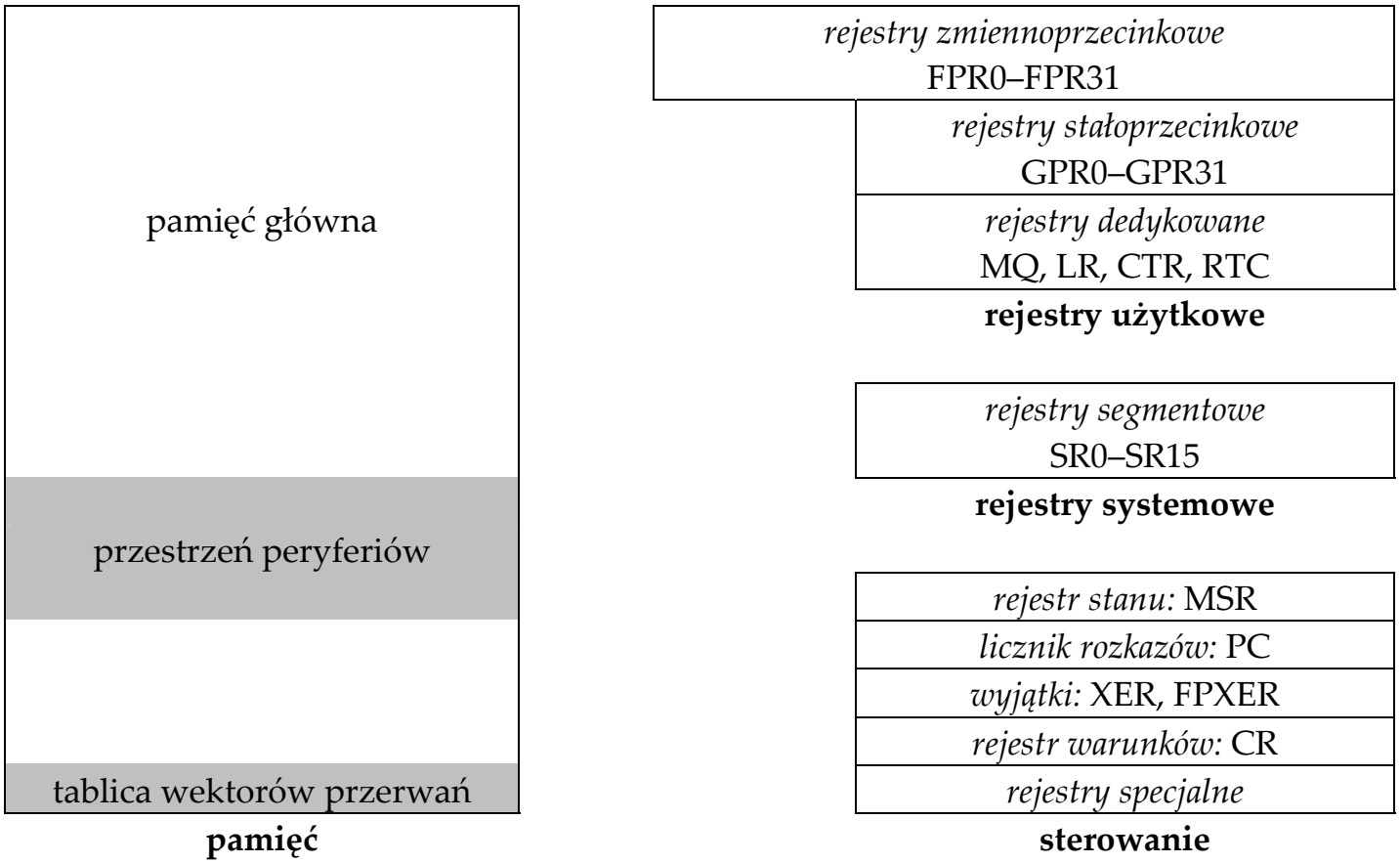
Separacja przestrzeni adresowych

- *fizyczna* – osobne układy dla różnych przestrzeni,
  rozróżnianie na podstawie rodzaju i 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*

Przestrzenie adresowe procesorów o architekturze IA-32e

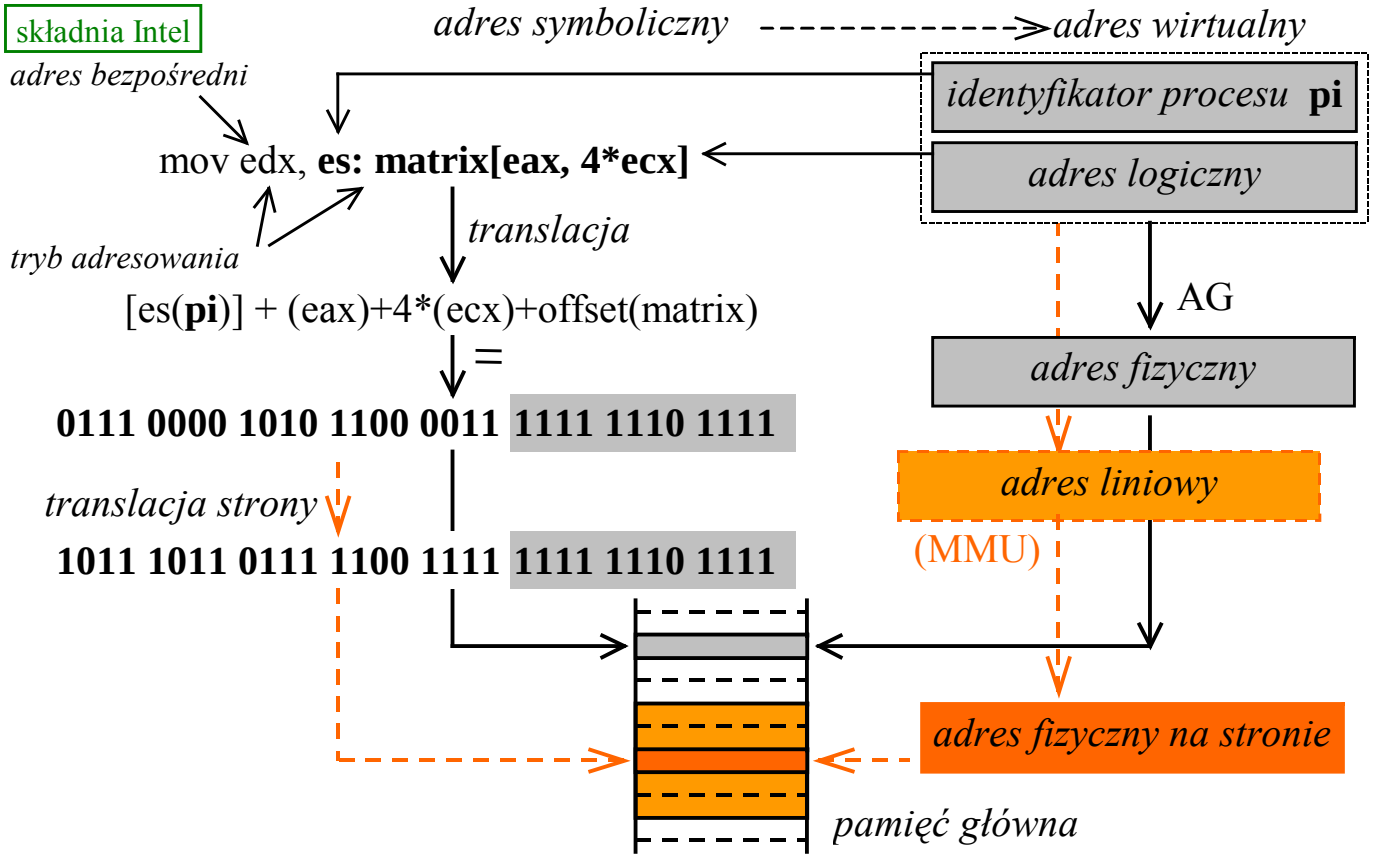


Przestrzenie adresowe procesorów PowerPC 601 (Motorola)



Tryby adresowania

sposób zamiany adresu symbolicznego na lokację w przestrzeni adresowej (IA-32)



Adresowanie bezpośrednie

- adresowanie zeroelementowe

kod rozkazu: | kod operacji | argument |

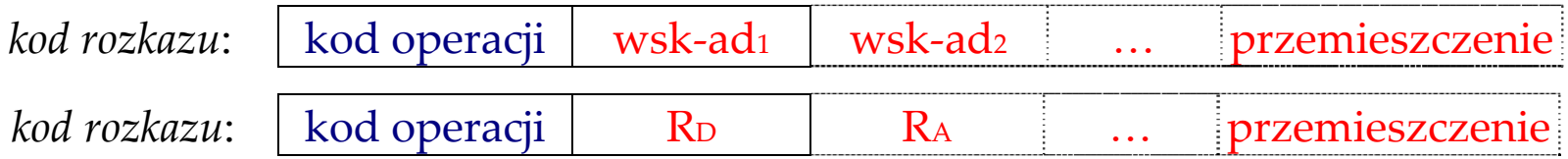
  - *błyskawiczne* (*quick*) – krótki kod danej w polu bitowym słowa kodu
    - adres argumentu – (*licznik rozkazów* : *pole kodu*)
  - *natychmiastowe* (*immediate*) –kod danej: rozszerzenie kodu rozkazu
    - adres argumentu = ++*licznik rozkazów*
  - *zwarte* (*compact*) – operandy domniemane

- adresowanie jednoelementowe

kod rozkazu: | kod operacji | adres argumentu |

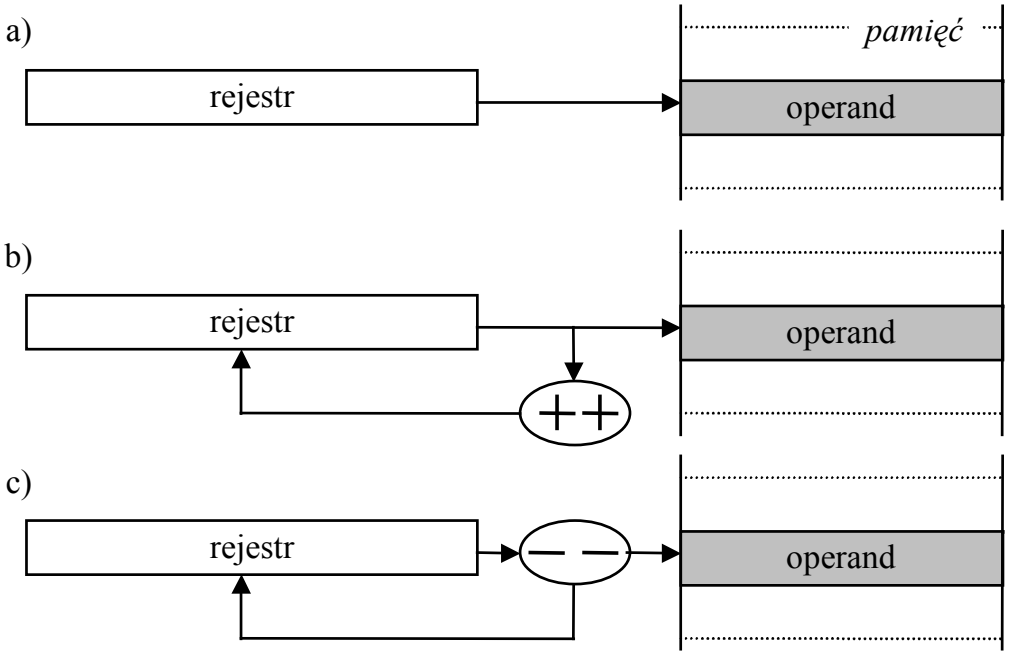
  - *bezwzględne* (*absolute*) – adres danej w polu słowa kodu rozkazu
    - adres adresu argumentu = ++ *licznik rozkazów*
  - *rejestrowe bezpośrednie* (*register direct*) – argument w rejestrze
    - adres argumentu – (*licznik rozkazów* : *pole kodu – numer rejestru*)

Adresowanie pośrednie



- 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 aktualizowany (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)	movl %eax, (%ebx)	move d3,(a5) (Motorola)
b)	pop %ecx (*niejawne*)	cmpm (a5)+, (a7)+ (Motorola)
c)	push %eax (*niejawne*)	abcd –(a3), –(a4)

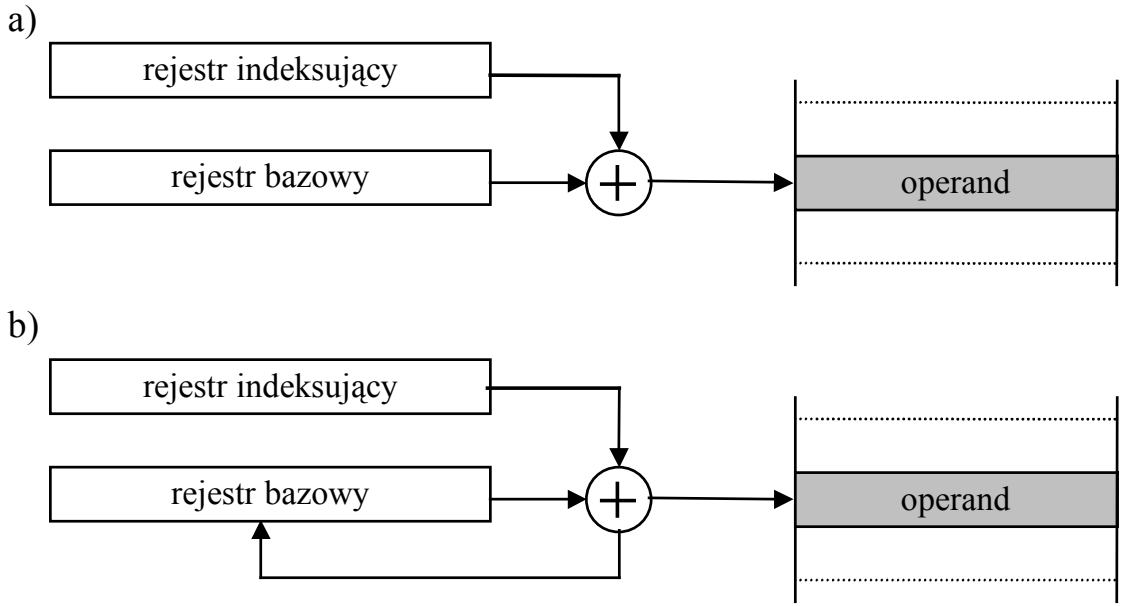
Adresowanie pośrednie wieloelementowe

składowe adresu pośredniego
- *baza* (*base*) – wskaźnik adresu bloku (zawartość rejestru procesora)
- *przemieszczenie* bazy (*offset*) – stała, adres odniesienia (wskaźnika) bloku
  $\rightarrow$ *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

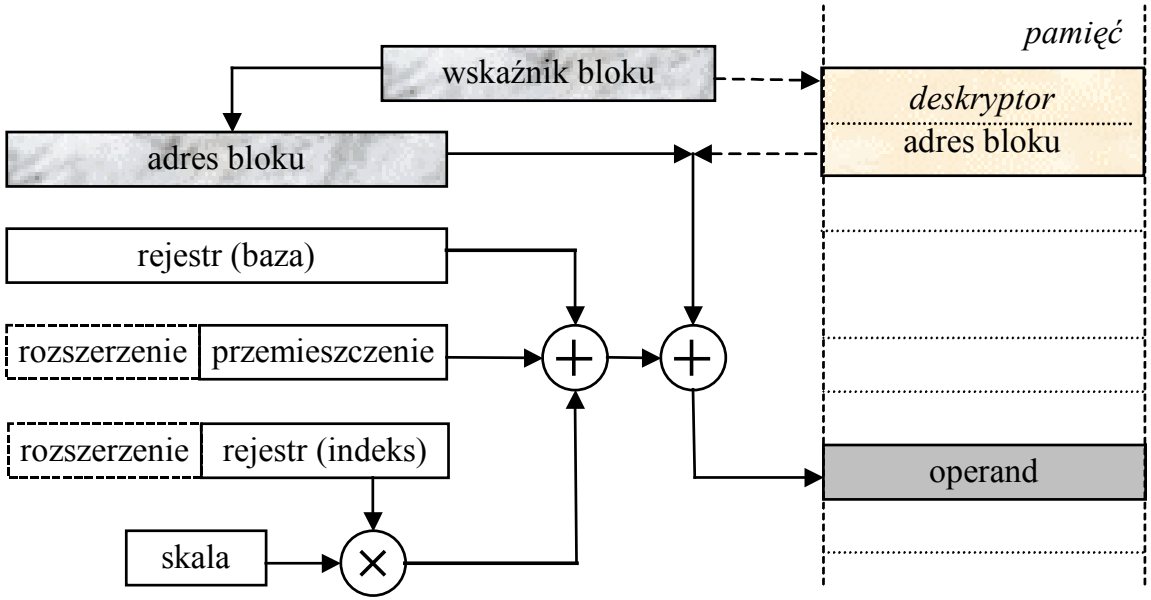


a) bazowo-indeksowe, b) bazowo-indeksowe z modyfikacją (PowerPC)

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

Adresowanie bazowo-indeksowe ze skalowaniem i deskryptorowe

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



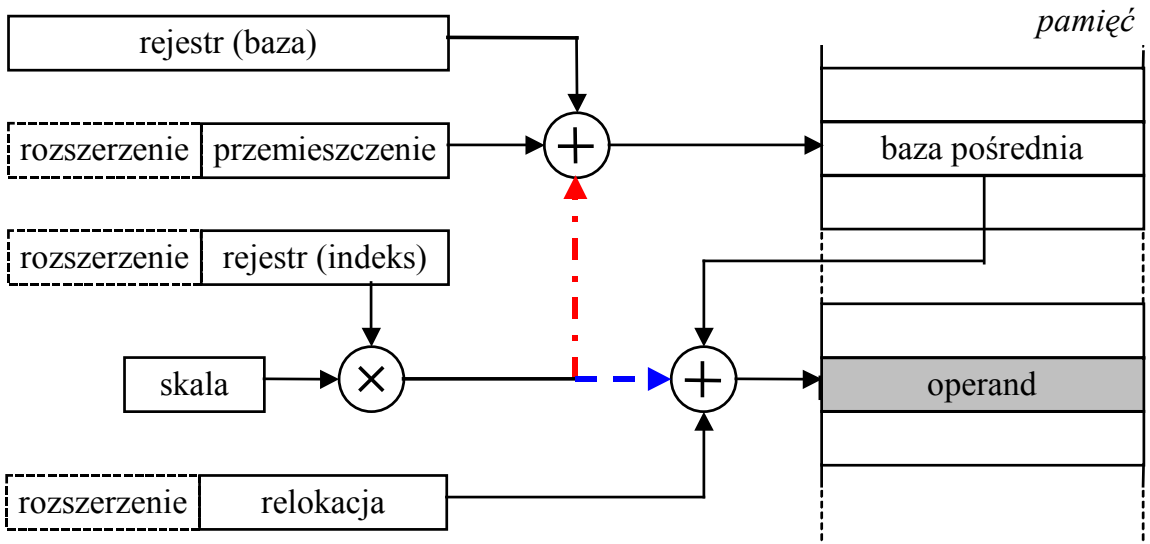
Adresowanie skalowane bazowo-indeksowe (i opisowe) (IA-32,...)

```
addl %eax, matrix(%ecx,%ebx,4)          ; przemieszczenie [baza, sk*indeks]
```

Adresowanie preindeksowe i postindeksowe

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

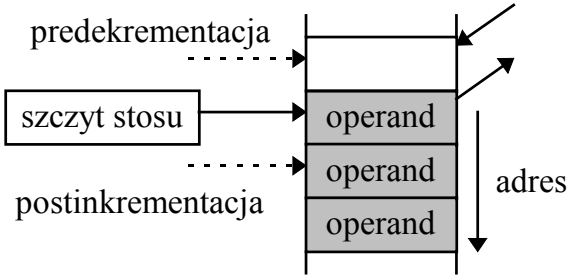
(*post*) LA = [(baza) + przemieszczenie] + (indeks) × skala + relokacja



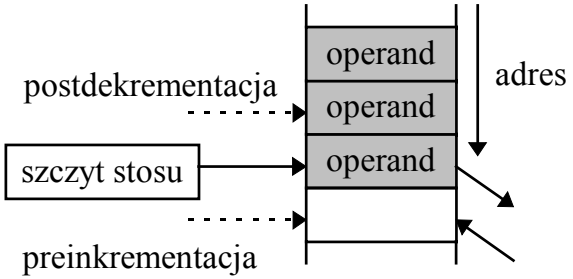
preindeksowe (-···-): move d3, ([baza, a4, d4.w], relokacja) (Motorola)

postindeksowe (----): move ([baza, a3], a2.w, relokacja), d5 (Motorola)

Adresowanie w obszarze stosu



stos budowany w kierunku adresów malejących

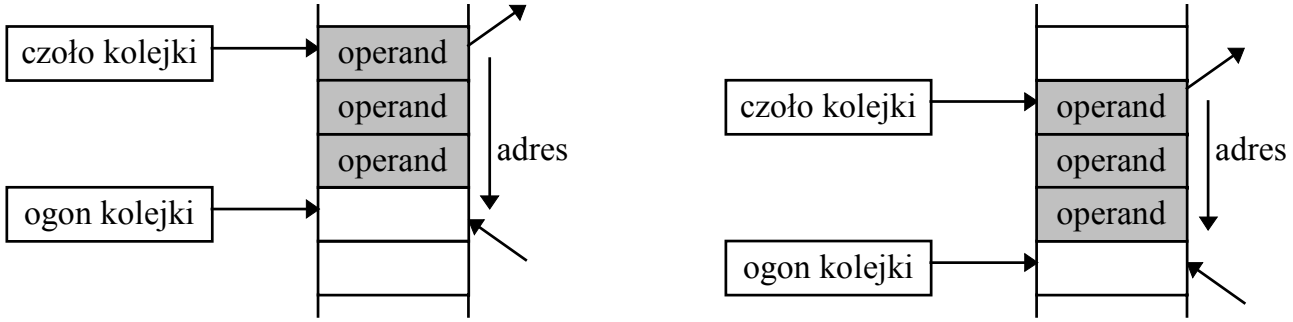


stos budowany w kierunku adresów rosnących

wskaźnik stosu (*stack pointer*, SP) – adres szczytu stosu

- lokacja szczytu stosu
  - ostatnia zajęta (wypełniona) – rozwiązanie elastyczne
  - najbliższa dostępna – rozwiązanie sztywne (ustalony rozmiar elementu)
- adresowanie stosu
  - domniemane – podczas wywołania procedur i funkcji (*adres powrotu*)
  - zamierzone – specjalne rozkazy (push / pop albo push / pull)

Adresowanie kolejki



Adresowanie w obszarze kolejki
konieczne dwa wskaźniki

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 (IA-32+) i Motorola 68K

- adresowanie stałych
  - addl $0x1245, %eax ; natychmiastowe (IA-32+)
  - adq –3, d5 ; błyskawiczne (Motorola 68K)
- adresowanie zwarte
  - imul %ebx ; domniemane: eax, edx (Intel)
- adresowanie rejestrowe bezpośrednie – ograniczone
  - subl %eax, %ecx ; (IA-32+)
  - add d3, d5 ; (Motorola 68K)
- adresowanie rejestrowe pośrednie
  - movl %eax, (%ebx, %ebp, 4) ; rejestry wskazane (IA-32+)
  - sub –(a3), –(a5) ; z automodyfikacją (Motorola 68K)
- adresowanie wieloelementowe
  - subl %eax, ta(%eax, %ecx, 4) ; rejestry wskazane (IA-32+)
  - cmp (a3)+, d5 ; z automodyfikacją (Motorola 68K)

Tryby adresowania w architekturze RISC

- adresowanie stałych
  - o adi.o r3, r5, –1 ; (PowerPC)
- adresowanie rejestrowe bezpośrednie – powszechne
  - o addo. r3, r7, r15 ; (PowerPC)
- adresowanie rejestrowe pośrednie – argumenty instrukcji *load* i *store*
  - o stw r5, r17 ; (PowerPC)
  - o lwzux r5, r17, r18 ; z aktualizacją (PowerPC)
  - o lwz r5, blok(r17) ; (PowerPC)
- adresowanie deskryptorowe
  - o mtsr sr7, r3 ; ładowanie r3 do rejestru segmentu sr7
  - o mtsr r3, r7 ; ładowanie r3 do rejestru segmentu
    ; wskazanego przez 4 wyższe bity r7

Adresowanie łańcuchów

ang. *Big Endian* *BE*
ważniejszy niższy
(*wysokokońcówkowy*)

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

ang. *Little Endian* *LE*
ważniejszy wyższy
(*niskokońcówkowy*)

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

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

Motorola 68K	BE
notacja	*adres*
tekst dc „drab"	adres tekst = = adres 'd'

Intel x86/Pentium	LE
notacja	*adres*
tekst dw „drab"	offset tekst = = offset 'b'

Czasowe aspekty adresowania – wyrównanie adresów

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)

wyrównanie adresu(ang. *alignment)*

– umieszczenie obiektu począwszy od adresu podzielnego przez jego rozmiar