

AKO OPRACOWANIE PYTAŃ Z E-NAUCZANIA

A1. Omówić podstawowe zasady wykonywania programu przez procesor

Trochę niejasne pytanie, ale chyba chodzi o cykl rozkazowy CPU

Ładowanie Rozkazu z pamięci -> Dekodowanie rozkazu -> ustawienie EIP, żeby wskazywało na następny rozkaz -> obliczenie adresu argumentu -> wykonanie rozkazu -> zapisanie wyniku do rej. lub pamięci



Tak wygląda uproszczony diagram z początku wykładu później po dekodowaniu rozkazu wspomniano o fazach:

- **Ustawiania EIP na kolejny rozkaz**
- **Wyznaczania adresu argumentu**

Dla wielu rozkazów nie potrzeba wszystkich faz, np przesłanie `mov eax, [ebp]` nie wymaga ALU

A2. Ile linii adresowych potrzebnych jest do adresowania pamięci 64 GB?

$$2^{32} = 4\text{GB}$$

$$4 \cdot 16 = 64 \rightarrow 2^{32} \cdot 2^4 = 2^{36}$$

Potrzeba 36 linii adresowych

A3. Porównać własności różnych rodzajów pamięci stosowanych w komputerach.

1. **Rejestry** - najszybsza pamięć w komputerze, jest najdroższa i jest jej najmniej, znajduje się na procesorze, służą do tymczasowego trzymania wyników obliczeń, adresów. Stanowią najwyższy szczebel w hierarchii pamięci.
2. **Cache** - Inaczej pamięć kieszeniowa, najczęściej typu **SRAM** dzieli się na on-chip i off-chip

Pamięci dynamiczne są zbyt wolne dla współczesnych CPU, przy dostępie występują stany oczekiwania. Pamięć cache zawiera pewną ilość obszarów z danymi (tzw linii wierszy), które służą do kopiowania bloków **pamięci operacyjnej**.

Może przechowywać rozkazy i dane.

Typowy blok zawiera 4-64 bajty.

W trakcie wykonywania rozkazów CPU szuka najpierw rozkazów i danych w pamięci podręcznej:

- hit
- miss (granica opłacalności to 20%, aktualnie dąży się do 5%)

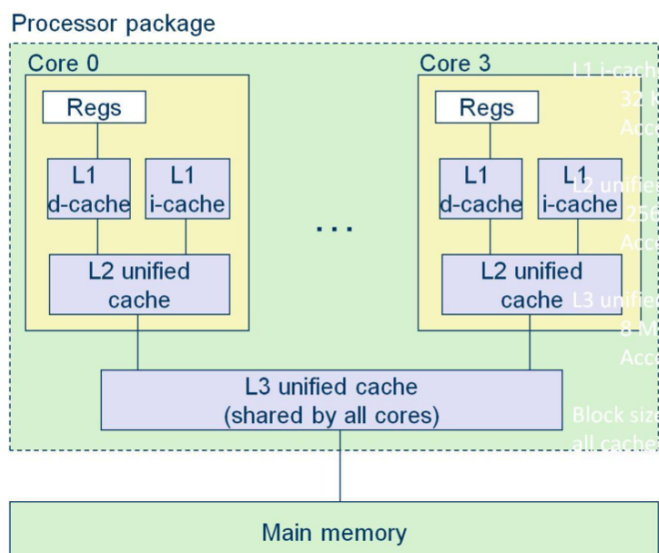
W i7 wygląda to tak:

(L0 Rejestry)

L1. Cache on chip, osobno dla każdego rdzenia (zintegrowana z rdzeniem) z podziałem osobno na dane i adresy

L2. Cache on chip, osobno dla każdego rdzenia (zintegrowana z rdzeniem) dla rozkazów i danych razem

L3. Cache off chip, wspólny dla wszystkich rdzeni



Zapewnienie spójności cache i ram:

1. Zapis przez (**write-through**) - zapis do ram po każdym zapisie w cache (wolne)
2. Zapis z opóźnieniem (**write back**) - zamiast zapisu do ramu zmienia się tylko **bit stanu**, trzeba aktualizować jeśli więcej wyjątków korzysta z tej pamięci

Pamięć SRAM (Przerzutnik dwustanowy)

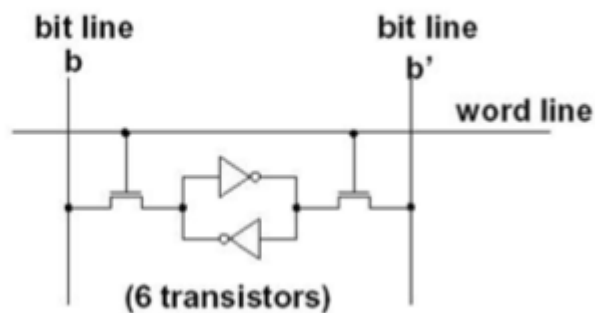
Zalety:

- Szybki dostęp do danych
- krótszy czas cyklu odczytu (nie trzeba odświeżać)
- wysoka odporność na zakłócenia

Wady:

- Koszt
- Niski stopień scalenia
- Duży pobór mocy

Pamięć statyczna SRAM



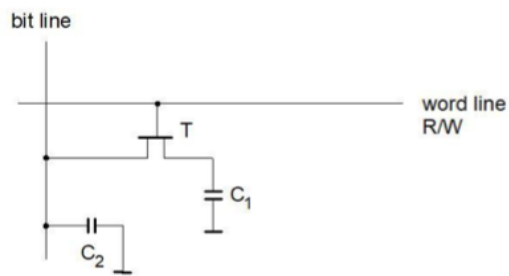
3. **Pamięć operacyjna** zazwyczaj DRAM (tranzystor i kondensator). Przechowywane są w niej aktualnie wykonywane programy i ich dane, wyniki. Ponieważ jest to pamięć dynamiczna to przed kolejną operacją odczytu trzeba ją odświeżać

Zalety:

- niski koszt
- względna energooszczędność
- duży stopień scalenia

Wady:

- podatność na zakłócenia
- czas dostępu
- czas cyklu odczytu
- konieczność odświeżania



FPMRAM -> SDR -> DDR -> DDR2

Jest bardzo wrażliwa na zakłucenia:

- dodatkowy bit (parity bit)
- w komp powyżej 1gb ram pełniących odpowiedzialne funkcję stosuje się pamięć ECC (Error checking and correction)

4. **Pamięć masowa** - np dysk twardy, bez dostępu do konkretnych bajtów

5. **ROM** -

- pamięć nieulotna
 - program inicjalizujący pracę komputera
1. ROM - zawartość w trakcie produkcji
 2. PROM (programmable ROM) jednokrotnie przez usera
 3. EPROM (erasable PROM) możliwość usuwania przy pomocy UV
 4. EEPROM (electrically EPROM)
 5. FLASH - pamięci błyskowe, możliwe jest programowanie całych bloków
 6. NVRAM połączenie SRAM z EEPROM

A4. W jaki sposób zmienia się zawartość wskaźnika instrukcji EIP w procesorach rodziny x86 w trakcie wykonywania różnych typów rozkazów?

1. Rozkazy nie sterujące
 $EIP = EIP + \text{długość rozkazu}$
2. Rozkazy sterujące
 - Skoki warunkowe np jmpe, jmpb
 - skoki bezwarunkowe np jmp etykieta
 - call, int
 - pośrednie/bezpośrednie np jmp dword ptr zmienna
 - zaliczyłbym tutaj też loop

A5. Scharakteryzować grupę instrukcji procesora określanych jako operacje bitowe.

ROL, ROR
 SHL, SHR

SAR, SAL
RCL, RCR
BT, BTR, BTS
AND, OR, XOR, MOT

A6. Wyjaśnić zasady działania układów DMA w komputerze.

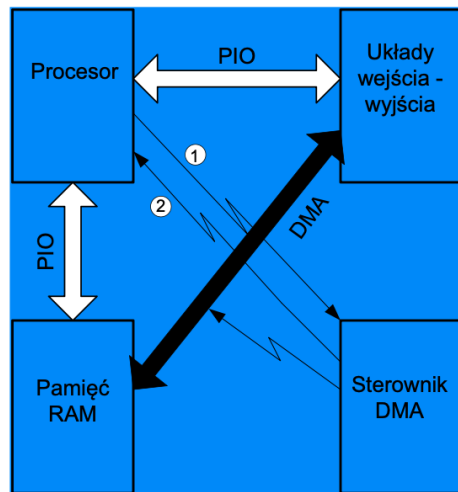
DMA (Direct Memory Access) - Technika przesyłania danych z pamięci głównej do urządzeń (lub odwrotnie) z pominięciem CPU.

Trzeba jedynie odpowiednio zainicjalizować układ DMA.

Po przesłaniu wszystkich bajtów DMA wywołuje przerwanie sprzętowe sygnalizując koniec przesyłania.

Moduł DMA potrzebuje takich informacji:

- rodzaj operacji (zapis, odczyt)
- adres urządzenia I/O
- adres obszaru RAM przewidzianego do odczytania/zapisania
- liczba bajtów



DMA używane są powszechnie do szybkich urządzeń np dyski, sterowniki usb
Istotne trudności w korzystaniu z DMA pojawiają się gdy PC ma pamięć podręczną.

B1. Omówić różne rodzaje kodowania liczb binarnych w komputerze.

1. NKB - zwykły dwójkowy
2. U2 (ze znakiem), najstarszy bit to liczba ujemna
Fajny sposób na konwersję NKB -> U2 (gdzie liczba nkb jest naszą liczbą, ale bez minusa)
Lecimy od tyłu (od najmłodszego bitu) aż pojawi się jedynka, zostawiamy ją w spokoju, a wszystkie bity za nią negujemy
np:
 $4 = 0100$
 $-4 = 1100$
3. ZM najstarszy bit jest znakiem
4. BCD - 4 bity kodują cyfry

5. Zmiennoprzecinkowe:
- np **float** 32 bity. niejawną jedynką
 - Z | 8 bitów wykładnika | 23 bity mantysy
 - <https://www.h-schmidt.net/FloatConverter/IEEE754.html>
 - double** 64 bity, niejawną jedynką
 - Z | 11 | 52
 - Format 80-bitowy** - jawna jedynka, wykorzystywany przez koprocessor w x86
 - Z | 15 | 64

B2. W jaki sposób w procesorach zgodnych z architekturą x86 sygnalizowane jest wystąpienie nadmiaru w operacjach dodawania, odejmowania, mnożenia i dzielenia na liczbach stałoprzecinkowych?

1. Dodawanie i odejmowanie liczb w NKB
Odejmowanie jest realizowane przez CPU jak dodawanie liczby przeciwnej.
CF zostaje ustawiony na 1, jeśli wystąpił nadmiar.
CF jest wyznaczone jako wartość przeniesienia przy dodawaniu dwóch ostatnich bitów
2. Dodawanie i odejmowanie liczb w NKB
Ustawiana jest OF
OF jest wyznaczana jako XOR CF i przeniesienia na ostatni bit
3. Mnożenie
Nadmiar nigdy nie wystąpi
4. Dzielenie
Nadmiar występuje gdy wyniki, lub reszta nie mieści się w przeznaczonych na nie rejestrach, albo gdy dzielimy przez 0. Zgłaszany jest wtedy wyjątek procesora (FAULT), więc można zamaskować

B3. Na czym polegają różnice w sposobie przechowywania liczb w pamięci znane jako mniejsze niżej (ang. little endian) i mniejsze wyżej (ang. big endian)?

LE - mniejsze części liczby na niższych adresach, zazwyczaj stosowane
BE - na odwrót, większe części liczby na mniejszych adresach

B4. Omówić technikę porównywania liczb stałoprzecinkowych stosowaną w architekturze x86.

CMP <1>, <2> - jest to instrukcja porównująca (nie wpisująca nigdzie wyniku odejmowania), która ustawia odpowiednio znaczniki ZF, OF, CF
OF dodatkowo tylko w liczbach ze znakiem

SUB robi dokładnie to samo, ale zapisuje wynik
Liczby można porównywać również przy użyciu instrukcji bitowych np. and

B5. Dlaczego procesor w trakcie wykonywania rozkazu dodawania ADD ustawia jednocześnie dwa znaczniki nadmiaru CF i OF ?

Dlatego, że instrukcja add działa tak samo dla liczb w U2 i NKB, ale inaczej powstaje nadmiar i te dwie opcje rozróżnia OF i CF.

Procesor nie wie na jakim formacie wykonuje rozkaz, więc rolą programisty jest interpretacja flag, cpu wyznacza dwie

B6. Czym różni się rozkaz CALL od rozkazu INT?

CALL - służy do wywołania procedur, zapisuje na stosie ślad - adres wskazujący na rozkaz, który ma być wykonany zaraz po procedurze

Funkcje kończą się rozkazem RET, wpisuje on do EIP (wskaźnika instrukcji) ślad

INT - stosujemy przy wywołaniu podprogramu systemowego przy pomocy tablicy wektorów przerwań. Zapisuje on ślad w postaci IP, CS, i FLAGS (IP w połączeniu z cs daje adres kolejnej instrukcji). Powrót sygnalizują się rozkazem IRET

B7. Zadaniem poniższej sekwencji rozkazów jest zwiększenie o 1 liczby całkowitej znajdującej się rejestrach RDX:RAX. Uzupełnić brakujący argument drugiego rozkazu

```
add rax, 1  
adc rdx, 0
```

Przy odejmowaniu jest taki rozkaz **SBB**

B8. W jakich przypadkach (w architekturze x86) zamiast rozkazów mnożenia i dzielenia można zastosować rozkazy SAL i SAR ?\

SAL - Gdy mnożymy przez potęgę 2. Trzeba uważać, czy CF = 1, bo to oznacza, że nasza liczba może się zmniejszyć zamiast zwiększyć

SAR - Dzielenie przez potęgę 2, powiela on bit znaku, więc rozkaz nadaje się też do działań na liczbach w U2

Rozkazy te są znacznie szybsze niż standardowe dzielenie/mnożenie

B9. W jakiej sytuacji, w trakcie wykonywania rozkazu sterującego (skoku), procesor ignoruje zawartość pola adresowego tego rozkazu?

Gdy warunek nie zostanie spełniony. Wtedy $EIP = EIP + d\ell \text{ rozkazu}$

B10. Wyznaczyć wartość dziesiętną 32-bitowej liczby zmiennoprzecinkowej (format float)

Z | 8 bitów wykładnika | 23 b mantysy

0 100 0000 1 111 1000 0000 0000 0000 0000

Wykładnik = $129 - 127 = 2$

Trzeba pamiętać o niejawniej jedynce

$1.1111 \cdot 2^2 = 111.11 = 7.75$ (w dziesiętnym)

B11. Co oznacza termin wartości specjalne używany w kontekście koprocatora arytmetycznego?

Są to wartości, które nie wynikają z definicji formatu zmiennoprzecinkowego, ale da się je zapisać na koprocesorze. Należą do nich:

1. +/- 0 - Wykładnik i mantysa to same zera
2. +/- nieskończoność - Wykładnik same jedynki, mantysa same zera
3. Nan (Not a number) - Wynik niedozwolonej operacji np pierw z -1. Wykładnik same 1, mantysa nie same 1
4. Liczby z niedomiarem, Wykładnik same 0, Mantysa nie same 0

B12. Czym różnią się rozkazy koprocatora arytmetycznego: FLD i FST?

Rozkaz **FLD** ładuje liczbę zmiennoprzecinkową na wierzchołek stosu koprocatora z pamięci, lub ze stosu koprocatora

Rozkaz **FST** przesyła z wierzchołka stosu koprocatora liczbę zmiennoprzecinkową do pamięci, lub na inny rejestr koprocatora

B13. Zadaniem poniższej sekwencji rozkazów jest wpisanie liczby 27.0 na wierzchołek stosu rejestrów koprocatora arytmetycznego. Uzupełnić brakujący rozkaz

push dword ptr 27

FLD DWORD PTR [ESP] ; bo całkowita

add esp, 4

B14. Wyjaśnić w jakim celu zdefiniowano nieliczby (NaN) w koprocesorze arytmetycznym.

Złożone działania numeryczne trwają czasami bardzo długo, godziny, czasem dni.

Wystąpienie nadmiaru, albo niedomiaru nie powinno powodować załamania programu. Praktyka pokazuje, że wyniki pośrednie z nadmiarem/niedomiarem mają często mały wpływ na wyniki końcowe.

B15. Do jakiej klasy wartości specjalnych liczb zmiennoprzecinkowych należy zaliczyć niżej podaną wartość typu float?

1	11111111	000000000000000000000000
---	----------	--------------------------

Jest to minus nieskończoność - znak to minus, wykładnik to same jedynki, a mantysa same zera

B16. Poniżej podano reprezentację binarną dwóch 32-bitowych liczb binarnych zmiennoprzecinkowych (format float) x i y. Ile wynosi iloraz x/y tych liczb ? Wynik podać w postaci liczby dziesiętnej (trzy cyfry po kropce)

x	0	01111111	000000000000000000000000
y	1	10000000	100000000000000000000000

x:

wykładnik: $127 - 127 = 0$

trzeba pamiętać o niejawnej "1"

+ 1.0 = 1 (dziesiętnie)

y:

znak to minus

wykładnik: $128 - 127 = 1$

11.0 = 3 (dziesiętnie)

-3

-0.333

B17. Omówić podstawowe zasady kodowania rozkazów procesora.

Producenci przygotowują instrukcję, przyporządkowują im konkretne ciągi zer i jedynek
Ciąg operacji dwuargumentowych powinien zawierać:

1. kod operacji
2. położenie pierwszego operandu
3. położenie drugiego operandu
4. informację dokąd przesłać wynik
5. Gdzie znajduje się kolejny wynik (Jeśli nie używa się wskaźnika instrukcji)

Konstruktorzy zwracają szczególną uwagę na długość instrukcji dlatego często wprowadza się ograniczenia takie jak np:

- wynik operacji wpisywany jest w miejsce pierwszego operandu
- Co najwyżej jeden argument może wskazywać na pamięć