1. Podaj przykłady rozkazów procesora: operacji arytmetycznych, logicznych, skoków – warunkowych i bezwarunkowych, dostępów do pamięci; użyj jednej ze stosowanych notacji i przykładowego procesora (istotna jest idea notacji i rozkazu, nie ścisła poprawność składniowa)

2. Podaj przykłady skalarnych i wektorowych rozkazów procesora – operacji arytmetycznych i dostępów do pamięci; użyj jednej ze stosowanych notacji i przykładowego procesora (istotna jest idea notacji i rozkazu, nie ścisła poprawność składniowa)

3. Podaj przykłady rozkazów dostępu do pamięci ze złożonym adresowaniem, wyjaśnij rolę poszczególnych parametrów

4. Czym jest blok podstawowy w kodzie asemblera? Jak wyszukiwać bloki podstawowe? Jaka jest rola bloków podstawowych w procesie optymalizacji?

5. Jaka jest zasada przetwarzania potokowego?

6. Porównaj pod kątem wydajności przetwarzanie rozkazów przez procesor (rdzeń): bez potokowości i z potokowością

7. Czym jest parametr CPI, a czym IPC? Jak wartość maksymalna tego ostatniego zależy od architektury procesora?

8. Wymień i krótko scharakteryzuj kilka zaawansowanych cech współczesnych procesorów (rdzeni) zwiększających wydajność klasycznego procesora potokowego.

9. Jak obliczać wydajność maksymalną potoków przetwarzania procesora? Jakie są ograniczenia praktycznego użycia ogólnych wartości IPC i CPI? Jak praktycznie oblicza się maksymalną wydajność rdzenia, mikroprocesora, komputera dla operacji zmiennoprzecinkowych?

10. Czym różni się opóźnienie przy wykonywaniu rozkazów przez procesor od (maksymalnej) przepustowości? Podaj przykład kodu, w którym o czasie wykonania będzie decydować opóźnienie (*latency*) przy wykonywaniu przez potoki przetwarzania konkretnej operacji arytmetycznej. Jakich wartości CPI należy się spodziewać w takim przypadku? Jak przełoży się to na wydajność w Gflop/s?

11. Podaj przykład kodu, w którym o czasie wykonania będzie decydować przepustowość (*throughput*) przy wykonywaniu przez potoki przetwarzania konkretnej skalarnej operacji arytmetycznej. Jakich wartości CPI należy się spodziewać w takim przypadku? Jak przełoży się to na wydajność w Gflop/s? Jak maksymalizować rzeczywistą przepustowość (średnie IPC) przy wykonaniu programu?

12. Jak skoki, warunkowe i bezwarunkowe, zaburzają przetwarzanie potokowe? W jaki sposób można uniknąć opóźnień przetwarzania potokowego w takiej sytuacji?

13. Jak obliczyć współczynniki CPI i IPC dla konkretnego wykonania programu? Jakich narzędzi można do tego użyć? Jakie są ograniczenia i trudności z interpretacją dla pomiaru CPI i IPC standardowych programów?

14. Podaj przykład kodu, w którym o czasie wykonania będzie decydować przepustowość (*throughput*) przy wykonywaniu przez potoki przetwarzania konkretnego rozkazu wektorowego odpowiadającego operacji arytmetycznej. Jakich wartości CPI należy się spodziewać w takim przypadku? Jak przełoży się to na wydajność w Gflop/s?

15. Czym są zależności danych między rozkazami wykonywanymi przez procesor? W jaki sposób zaburzają przetwarzanie potokowe? W jaki sposób można uniknąć opóźnień przetwarzania potokowego w takiej sytuacji?

16. Jak wygląda faza pobierania i dekodowania oraz przekazywania do wykonania rozkazów przez współczesne procesory?

17. W jaki sposób procesor może przeprowadzać przewidywanie skoków (rozgałęzień)?

18. Jak działa pamięć wirtualna ze stronicowaniem?

19. Opisz proces translacji adresów wirtualnych na fizyczne. Czym jest tablica stron (*page table*)?

20. Co to jest TLB (*translation lookaside buffer*)? Kiedy następuje intensywne użycie TLB?

21. Czym jest mniejszy błąd strony (*minor page fault*), a czym większy błąd strony (*major page fault*)

22. Czym jest szamotanie (*thrashing*)? Jak można go unikać?

23. Jaki jest w przybliżeniu czas wykonania pojedynczej operacji arytmetycznej przez procesor, a jaki czas pobrania pojedynczej zmiennej z pamięci DRAM, zakładając izolowane operacje (czyli czasy typowe dla opóźnienia, *latency*)

24. Jaki jest w przybliżeniu czas wykonania pojedynczej operacji arytmetycznej przez procesor, a jaki czas pobrania pojedynczej zmiennej z pamięci DRAM, zakładając długie sekwencje optymalnie wykonywanych operacji (czyli czasy typowe dla przepustowości, *throughput*)

25. Podaj przykład kodu, w którym występuje zależność danych dla kolejnych iteracji pętli. W jaki sposób wpływa ona na wydajność przetwarzania.

26. Podaj przykład kodu, w którym o czasie wykonania będzie decydować opóźnienie (*latency*) pobierania danych z pamięci. Jakie cechy kodu decydują o tym, że procesor nie jest w stanie wykorzystać sprzętowych technik ukrywania opóźnienia (scharakteryzuj dla konkretnych technik ukrywania opóźnienia)?

27. Podaj przykład kodu, w którym o czasie wykonania będzie decydować przepustowość (*throughput*) pobierania danych z pamięci. Jakie cechy kodu decydują o tym, że procesor jest w stanie wykorzystać sprzętowe techniki ukrywania opóźnienia (scharakteryzuj dla konkretnych technik ukrywania opóźnienia)?

28. Jak działa pamięć podręczna (*cache memory*)? Omów na przykładzie jednopoziomowej pamięci podręcznej.

29. Podaj przykład kodu, który może posłużyć do ustalenia rozmiaru pamięci podręcznych poszczególnych poziomów w mikroprocesorze. Omów zasadę pomiaru.

30. Podaj przykład kodu, który może posłużyć do ustalenia rozmiaru linii pamięci podręcznej L1 w rdzeniu mikroprocesora. Omów zasadę pomiaru.

31. Skąd bierze się pozytywny wpływ istnienia pamięci podręcznej na wydajność pracy procesora? Jakie są typy lokalności odniesień?

32. Jak działa pamięć podręczna bezpośrednio odwzorowana (*direct mapped*) w odróżnieniu od pamięci w pełni skojarzeniowej (*fully associative*)

33. Co to jest drożność pamięci podręcznej? Podaj schemat działania dwudrożnej pamięci sekcyjno-skojarzeniowej (*2-way set associative*).

34. Kiedy występuje chybienie wymuszone (*compulsory miss*) w pamięci podręcznej?

35. Kiedy występuje chybienie pojemnościowe (*capacity miss*) w pamięci podręcznej?

36. Kiedy występuje chybienie konfliktowe (*conflict miss*) w pamięci podręcznej?

37. Jakie są mechanizmy ukrywania opóźnienia przy dostępach do pamięci? Jak pisać kod maksymalizujący wydajność pamięci?

---

38. Jakie zasoby są wspólne, a jakie odrębne dla wątków tego samego procesu?

39. Co to jest przełączanie kontekstu? W jaki sposób opóźnia działanie programu?

40. Czym jest sprzętowa jednoczesna wielowątkowość (*simultaneous multithreading*)? Jak reprezentowana jest na poziomie systemu operacyjnego? Jak wpływa na wydajność programów?

41. Czym różni się organizacja UMA i NUMA dostępu do pamięci? Jaki ma wpływ na wydajność?

42. Przedstaw strategie przypisywania wątków do rdzeni (procesorów logicznych dla SMT)

43. Omów strategie *write-through*, *write-back*, *write-allocate*, *no-write-allocate* zapisu danych do pamięci.

44. Scharakteryzuj protokoły utrzymywania zgodności pamięci podręcznej (*cache coherence protocols*) z unieważnianiem (np. protokół MESI)

45. Kiedy występuje chybienie spójnościowe (*coherency miss*) w pamięci podręcznej?

46. Co to jest, do czego służy i jak można uzyskać profil wykonania kodu?

47. Programy profilujące (profilery) – zasady działania, przykłady

48. Czas wykonania – zewnętrzny, CPU, użytkownika, systemowy, narzędzia systemowe zwracające czas wykonania – dla programów jednowątkowych i wielowątkowych

49. Czym są liczniki sprzętowe i do czego można je wykorzystać? Podaj przykłady zdarzeń sprzętowych (*hardware events*)

50. Narzędzia (systemowe i inne) korzystania z liczników sprzętowych

51. Na czym polega strategia optymalizacji przez usuwanie wąskich gardeł (*bottlenecks*)?

52. Jakie są typowe kroki podejmowane przy optymalizacji kodu ze względu na wydajność?

53. Co ogranicza wydajność w pojedynczym węźle obliczeniowym?

54. Co to jest diagram *roofline* i do czego służy? W jaki sposób uzyskiwane są proste na diagramie *roofline*?

55. Jak uzyskać poziomą linię na eksperymentalnym diagramie *roofline?* (czyli jak zaprojektować benchmark do maksymalnego wykorzystania mocy potoków przetwarzania)? Jakie warianty można rozważyć?

56. Jak uzyskać nachyloną linię na diagramie *roofline*? (czyli jak zaprojektować benchmark do pełnego wykorzystania możliwości układu pamięci i jak wykorzystać jego wyniki)

57. Jak obliczyć współczynnik s_{pm} dla programu? Uwzględnij warianty dla pamięci DRAM i pamięci podręcznych różnych poziomów.

58. Jak sprawdzić, korzystając z diagramu *roofline*, czy optymalizowany program dopuszcza jeszcze znaczące zyski wydajności?

59. Jaki jest wpływ zwiększania lokalności odniesień w kodzie na współczynnik intensywności arytmetycznej s_{pm} ?

60. Jakie optymalizacje mogą zwiększyć współczynnik intensywności arytmetycznej s_{pm} i dlaczego?

61. Jak wykorzystać diagram *roofline* do optymalizacji algorytmu?

62. Kiedy mówimy, że wydajność programu jest ograniczana przez wydajność pamięci? Zilustruj ten fakt korzystając z diagramu *roofline*

63. Kiedy mówimy, że wydajność programu jest ograniczana przez wydajność (potoków przetwarzania) procesora? Zilustruj ten fakt korzystając z diagramu *roofline*

64. Jakie jest znaczenie punktu przecięcia linii na diagramie *roofline*, odpowiadającego tzw. równowadze sprzętu (*machine balance*)?

65. Jakie dwa podstawowe czynniki (dwie podstawowe cechy wydajnościowe) wpływają na ostateczną bezwzględną wydajność programów równoległych? Podaj odpowiednie wzory dla wydajności wyrażanej w Gflop/s.

66. Jak w przybliżeniu można szacować czas przesłania pojedynczego komunikatu o *m* bajtach, pomiędzy dwoma węzłami sieci komputerowej?

---

67. Na czym polegają klasyczne optymalizacje *constant folding, copy propagation, strength reduction, common subexpression elimination*? Podaj przykłady optymalizacji i ich wpływ na wydajność

68. Na czym polegają klasyczne optymalizacje dotyczące pętli: *loop invariant code motion* i *induction variable simplification*? Podaj przykłady optymalizacji i ich wpływ na wydajność

69. Na czym polegają klasyczne optymalizacje dotyczące pętli: *loop fusion* i *loop fission*? Podaj przykłady optymalizacji i ich wpływ na wydajność

70. Na czym polegają klasyczne optymalizacje dotyczące pętli: *loop interchange* i *loop unrolling*? Podaj przykłady optymalizacji i ich wpływ na wydajność

71. Na czym polega optymalizacja *register blocking*? Podaj przykład optymalizacji i jej wpływ na wydajność

72. Na czym polegają optymalizacje: *software prefetching* i *software pipelining*? Podaj przykłady optymalizacji i ich wpływ na wydajność

73. Na czym polega optymalizacja *cache blocking*? Podaj przykład optymalizacji i jej wpływ na wydajność

74. Czym jest zjawisko fałszywego współdzielenia (*false sharing*)? Podaj przykład kodu, w którym może wystąpić zjawisko fałszywego współdzielenia (*false sharing*) przy wykonaniu wielowątkowym. Opisz jak przebiega wykonanie kodu w takim przypadku (w szczególności jak funkcjonuje pamięć podręczna) i jak można zmodyfikować kod, aby uniknąć fałszywego współdzielenia.

75. Co to jest przyspieszenie obliczeń i efektywność zrównoleglenia?

76. Czym różni się badanie skalowalności w sensie słabym od badania skalowalności w sensie silnym?

77. Jak tworzy się wykres przyspieszenia przeskalowanego (*scaled speed-up*)?

78. Kiedy mówimy, że program jest skalowalny w sensie słabym?

79. Jakie czynniki wpływają na wydajność programów równoległych (stanowią narzut wykonania równoległego)?

80. Jaki jest warunek dotyczący czasu wykonania równoważny liniowej skalowalności w sensie słabym?

81. Przedstaw i objaśnij tzw. równanie wydajności (*performance equation*). Co oznaczają i jak można optymalizować wartości czynników występujących we wzorze?

82. Omów pojecie narzutu czasowego obliczeń równoległych. Jak powinien zachowywać się narzut, żeby obliczenia były skalowalne w sensie silnym, a co wystarcza do skalowalności w sensie słabym?