Architektury systemów komputerowych

MIPS

Poniższa lista zawiera tematy projektów (lub projekcików) programistycznych do zrealizowania w asemblerze procesora MIPS.

Wybór zadania. Zadania są przeznaczone dla pojedynczych osób lub grup dwuosobowych (przy każdym zadaniu zaznaczone jest, czy może być one wykonywane przez zespół dwuosobowy). W jednej grupie ćwiczeniowej dane zadanie może być realizowane maksymalnie przez dwa zespoły (*zespołem* nazywamy też osobę samodzielnie rozwiązującą zadanie). Rezerwacji zadania można dokonać u swojego ćwiczeniowca (osobiście lub za pomocą poczty elektronicznej). W zgłoszeniu należy podać numer zadania oraz skład zespołu. Każdy zespół powinien składać się z członków tej samej grupy ćwiczeniowej (wyjątki są dopuszczalne tylko po wcześniejszym uzgodnieniu z prowadzącymi ćwiczenia). Pierwszy przydział zadań nastąpi w piątek, 23 maja. Zgłoszenia do pierwszego przydziału przyjmowane są do tego dnia, do godz. 11.00. Zgłoszenie może mieć formę listy preferencji. 23 maja, jeśli więcej niż jeden zespół wyrazi chęć rozwiązywania tego samego zadania, o przydziale zadecyduje średnia liczba punktów ćwiczeniowych zdobytych przez członków grupy (im wyższa, tym lepiej). Po tym terminie można nadal nadsyłać zgłoszenia, ale od tego momentu decyduje kolejność zgłoszeń.

Punktacja. Do minimum na bdb wliczamy 15 punktów. Każde zadanie warte jest od 8 do 40 punktów – liczba punktów podana jest przy treści zadania. Ocena dla zespołu dwuosobowego jest jedna, tzn. obaj członkowie grupy otrzymują jednakową liczbę punktów (liczba wskazana przy treści zadania jest maksymalną oceną dla każdgego członka zespołu). Przy ocenianiu programu, opócz poprawności działania, będzie brana pod uwagę elegancja i efektywność rozwiązania.

Oddawanie zadań. Zadania należy oddawać najpóźniej na ostatnich ćwiczeniach w semestrze. Szczegółowe zasady oddawania zadań ustalają prowadzący. Prowadzący ma prawo zażądać obecności całego zespołu podczas prezentacji rozwiązania i zapytać dowolnego z jego członków o dowolny kawałek kodu. Odpowiedź typu „niestety, kod pisałem dawno i nie pamiętam w tej chwili jak ten fragment działa i dlaczego tak jest napisany" może być podstawą niezaliczenia zadania danemu członkowi grupy. Dlatego do prezentacji rozwiązania należy się odpowiednio przygotować. Programy powinny działać w symulatorze SPIM.

Od tego momentu nie na każde ćwiczenia będzie przygotowywana lista zadaniowa (proszę obserwować moją stronę internetową, jeżeli do środy, do godz. 14 nie ma kolejnej listy, to znaczy, że już jej w danym tygodniu nie będzie). Obecność na ćwiczeniach bez listy jest nieobowiązkowa. Takie ćwiczenia przeznaczone będą na konsultacje w sprawie projektów lub oddawanie projektów. Szczegóły należy uzgodnić z prowadzącymi.

Zadania można oddawać oczywiście również wcześniej niz na ostatnich ćwiczeniach: na wcześniejszych ćwiczeniach (o ile nie ma na nie przygotowanej listy zadaniowej, lub lista ta zostanie rozwiązana szybko), podczas konsultacji prowadzących lub w innym uzgodnionym z prowadzącymi terminie).

TREŚCI ZADAŃ

1. **Sortowanie 1 (15 pkt., 1 os.).** Napisz procedury realizujące sortowanie liczb całkowitych metodami *quicksort* i *selection sort*. Argumentami procedur powinny być: adres pierwszej danej oraz liczba danych do posortowania. Napisz krótki program demonstrujący wykorzystanie procedur (wprowadzanie danych z klawiatury, wybór metody, sortowanie, wypisanie wyniku na ekran).
2. **Sortowanie 2 (15 pkt., 1 os.).** Napisz procedury realizujące sortowanie liczb całkowitych metodami *insertion sort* i *bubble sort*. Argumentami procedur powinny być: adres pierwszej danej oraz liczba danych do posortowania. Napisz krótki program demonstrujący wykorzystanie procedur (wprowadzanie danych z klawiatury, wybór metody, sortowanie, wypisanie wyniku na ekran).
3. **Wieże Hanoi (15 pkt., 1 os.).** Napisz program, który wczytuje z klawiatury liczbę N, a następnie wypisuje rozwiązanie (sekwencję ruchów) dla problemu wież Hanoi z N krążkami.
4. **Równania kwadratowe (8 pkt., 1 os.).** Napisz program rozwiązujący równanie kwadratowe. Współczynniki powinny być wczytywane z klawiatury.

5. **Wyznacznik (22 pkt., 1 os.).** Napisz program, który wylicza wartość wyznacznika złożonego z liczb całkowitych. Zadanie rozwiąż rekurencyjnie, rozwijając wyznacznik względem pierwszego wiersza. Zakładamy, że program ma działać poprawnie dla takich danych, dla których wyniki pośrednie mieszczą się w zakresie liczb 32-bitowych. Dane powinny być wczytywane z klawiatury.
6. **Sortowanie napisów (18 pkt., 1 os.).** Napisz program, który wczytuje z klawiatury liczbę N, a następnie N napisów, po czym sortuje je i wypisuje na ekran w kolejności leksykograficznej. Możesz założyć, że $N < 1000$ oraz, że każdy napis ma długość mniejszą od 20 i składa się wyłącznie z liter alfabetu łacińskiego.
7. **Zliczanie wzorca (15 pkt., 1 os.).** Napisz program, który wczytuje z klawiatury dwa łańcuchy znaków, a następnie informuje ile razy drugi z nich pojawia się w pierwszym jako spójny podciąg. W programie wyróżnij procedurę, która jako parametry dostaje adresy dwóch napisów.
8. **Zamiana wzorca (20 pkt., 1 os.).** Napisz program, który wczytuje z klawiatury trzy napisy, a następnie wypisuje na ekran napis, który powstaje przez zastąpienie w pierwszym napisie każdego wystąpienia drugiego napisu trzecim.
9. **Kody Hamminga (18 pkt., 1 os.).** Napisz program, który potrafi operować kodem Hamminga poprawiającym pojedyncze błędy. Wyróżnij procedurę, która interpretuje swój jedyny parametr jako kod ASCII pewnego znaku (liczy się 8 najmniej znaczących bitów) i zwraca kod Hamminga tego znaku (liczy się 12 najmniej znaczących bitów). Napisz również procedurę działającą w drugą stronę. W przypadku pojedynczego błędu druga procedura powinna zwrócić skorygowany znak i poinformować o błędzie w swoim drugim parametrze. Przygotuj krótki program demonstracyjny.
10. **Konwersja (20 pkt., 1 os.).** Napisz program konwertujący liczby całkowite pomiędzy różnymi systemami liczbowymi. Zakładamy, że podstawy systemów i same liczby podawane są z klawiatury. Program powinien obsługiwać systemy o podstawach od 2 do 16.
11. **Kalkulator (12 pkt., 1 os.).** Napisz program prostego kalkulatora całkowitoliczbowego. Program powininien wczytywać w pętli liczbę i znak działania i podawać aktualny wynik.
12. **64-bitowy Fibonacci (22 pkt., 1 os.).** Napisz program wyliczający n-tą liczbę Fibonacciego F_n. Program powinien działać poprawnie dla wszystkich n, dla których F_n mieści się na **64** bitach.
13. **Drzewa binarne (30 pkt., 2 os.).** Zrealizuj w MIPS-ie operacje na drzewach binarnych przeszukiwań: wstawianie, wyszukiwanie, sprawdzanie czy element jest w drzewie, usuwanie.
14. **Arytmetyka 64-bitowa (30 pkt., 2 os.).** Przygotuj zestaw procedur, realizujących operacje całkowitoliczbowe na liczbach 64-bitowych w reprezentacji uzupełnień do 2. Zrealizuj następujące operacje:
    (a) wypisywanie liczby w postaci binarnej, szesnastkowej i **dziesiątkowej**
    (b) dodawanie
    (c) odejmowanie
    (d) mnożenie
    (e) dzielenie

    Możesz korzystać ze wszystkich MIPS-owych rozkazów 32-bitowych.
15. **Arytmetyka zmiennopozycyjna (40 pkt., 2 os.).** Przygotuj zestaw procedur realizujących operacje zmiennopozycyjne (uproszczone IEEE 754, tzn. nie ma żadnych sytuacji wyjątkowych) na rejestrach całkowitoliczbowych. Nie wolno używać rozkazów i rejestrów koprocesora zmiennopozycyjnego. Zrealizuj następujące operacje:
    (a) wypisywanie liczby na konsolę w postaci: znak, mantysa, wykładnik.
    (b) wypisywanie liczby dziesiętnie, w postaci: $a,mmmmmm$, z zadaną jako parametr liczbą miejsc po przecinku
    (c) wyznaczanie wartości bezwzględnej
    (d) porównywanie
    (e) dodawanie
    (f) odejmowanie
    (g) mnożenie
    (h) dzielenie

16. **Obliczanie pierwiastka kwadratowego metodą Newtona (8 pkt., 1 os.).** Napisz program, który wylicza wartość pierwiastka kwadratowego z wczytanej liczby, korzystający ze wzoru rekurencyjnego Newtona: $x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2}$.
17. **ONP (18 pkt., 1 os.).** Napisz program, który wylicza wartość wczytanego wyrażenia w notacji ONP.
18. **a^n mod m (25 pkt., 1 os.).** Napisz procedurę, która wylicza $a^n \bmod m$. Procedura powinna działać dla dowolnych a, m i n mieszczących się na 32-bitach.
19. **Potęgowanie macierzy (15 pkt., 1 os.).** Napisz procedurę, która podnosi do n-tej potęgi macierz kwadratową o rozmiarach 2×2, wykonując $O(\log n)$ operacji. Wykorzystaj tą procedurę do wyznaczania k-tej liczby Fibonacciego (wylicz odpowiednią potęgę macierzy złożonej z trzech jedynek i jednego zera).
20. **Szyfr Cezara (10 pkt., 1 os.).** Napisz procedury, które potrafią kodować i rozkodowywać teksty przy użyciu *szyfru Cezara*. Przesunięcie powinno być parametrem procedury. Przygotuj, krótki program demonstracyjny.
21. **Wyznaczanie k-tego elementu (12 pkt., 1 os.).** Zaimplementuj algorytm Hoare'a wyznaczania k-tego największego elementu w zadanym ciągu liczb.
22. **Najdłuższy wspólny podciąg dwóch słów (15 pkt., 1 os.).** Zaimplementuj algorytm dynamiczny wyznaczający najdłuższy wspólny podciąg dwóch słów.
23. **Jednowymiarowa „gra w życie" (18 pkt., 1 os.).** Napisz program symulujący jednowymiarową „grę w życie" (wariant znanej gry Conway'a). Pojedyncze pokolenie opisujemy jako ciąg zer i jedynek długości 32. Zero oznacza komórkę martwą, jedynka - żywą. W następnym pokoleniu zmiany są następujące: komórka żywa, która ma 0,1 lub 3 sąsiadów umiera, komórka martwa, która ma 2 lub 3 sąsiadów ożywa. Sąsiadami komórki są komórki oddalone od niej o 1 lub 2 pozycje. Napisz procedurę, która interpretuje parametr otrzymany w rejestrze `$a0` jako opis jednego pokolenia i zwraca w rejestrze `$v0` opis kolejnego pokolenia. Program główny powinien wczytywać opis pierwszego pokolenia, a następnie cyklicznie generować i wypisywać kolejne pokolenia. Kolejne pokolenia wypisuj np. używając kropek na oznaczenie komórek martwych, a gwiazdek na oznaczenie komórek żywych.
24. **Liczby doskonałe (8 pkt., 1 os.).** Napisz procedurę, która sprawdza czy podana w rejestrze `$a0` liczba naturalna jest liczbą doskonałą. Przypominam, że liczba jest *doskonała* jeśli jest sumą wszystkich swoich dzielników właściwych.
25. **Rozkład na czynniki pierwsze (8 pkt., 1 os.).** Napisz procedurę, która rozkłada podaną w rejestrze `$a0` liczbę naturalną na czynniki pierwsze.
26. **Sito Eratostenesa (8 pkt., 1 os.).** Napisz program, który dla zadanego n wypisuje na ekran wszystkie liczby pierwsze mniejsze od n, korzystając z sita Eratostenesa.
27. **Liczba dni pomiędzy dwiema datami (15 pkt., 1 os.).** Napisz program, który dla podanych z klawiatury dwóch dat wylicza liczbę dni, które upłynęły pomiędzy nimi.
28. **Trójkąt Pascala (8 pkt., 1 os.).** Napisz program, który wypisuje na ekranie trójkąt Pascala o zadaniej liczbie wierszy (rozsądnej...).