Architektury systemów komputerowych

Lista 6

$x_6 = 6$ (minimum na bdb)

Lista 6 przeznaczona jest na ćwiczenia w tygodniu 7-11 kwietnia (o ile prowadzący ćwiczenia w poszczególnych grupach nie ustalą inaczej). Zadania polegają na napisaniu programów w asemblerze procesora MIPS. Wszystkie programy powinny wykonywać się na symulatorze MARS.

Zasady oddawania zadań w moich grupach: najpóźniej w dniu poprzedzającym ćwiczenia, do godz. 22.00, należy przysłać mi mailem (na adres `kiero@cs.uni.wroc.pl`, temat wiadmości: 'MIPS') skompresowany programem zip plik o nazwie 'INa.zip' (gdzie 'I' to pierwsza litera imienia, a 'Na' dwie pierwsze litery nazwiska autora), zawierający pliki o nazwach 'INai.asm' (gdzie 'INa' jest jak wyżej, a 'i' to numer zadania) z rozwiązaniami poszczególnych zadań. Na ćwiczenia (które odbędą się najprawdopodobniej w sali 141) należy przyjść ze standardową deklaracją z zaznaczonymi numerami przysłanych zadań. Wybrane osoby będą prezentowały swoje rozwiązania zadań (z mojego laptopa, na ekranie telewizora lub przy pomocy rzutnika). Podczas prezentacji należy przedstawić działanie programu, zaprezentować, omówić jego kod i odpowiedzieć na ewentualne pytania prowadzącego lub innych studentów (przed ćwiczeniami należy dobrze przypomnieć sobie swoje rozwiązania; odpowiedź typu: 'pisałem to parę dni temu i już nie pamiętam co robi ten fragment kodu', czy 'nie wiem jak działa ten rozkaz' kwalifikuje się na „grzybka").

1. Napisz program, który wylicza sumę oraz maksimum z tablicy liczb całkowitych, której adres zadany jest w rejestrze $a0, a rozmiar w rejestrze $a1. Wynik umieść w rejestrze $v0.

2. Napisz program, który sortuje tablicę liczb całkowitych metodą *quicksort* (założenia o tablicy jak wyżej).

3. Napisz program, który zlicza ile jedynek znajduję się w bitowej reprezentacji rejestru `$s0`.

4. Załóżmy, że rejestr $a0 przechowuje **dwie** liczby szesnastobitowe: liczbę a na swoich bardziej znaczących bitach oraz liczbę b na swoich mniej znaczących bitach. Obie zakodowane w reprezentacji uzupełnień do 2. W asemblerze MIPS napisz program, który umieści w rejestrze $v0 dwie liczby szesnastobitowe: na bardziej znaczących bitach liczbę $a+b$, a na mniej znaczących – liczbę $a-b$. Zakładamy, że błędy przepełnienia nie wystąpią (nie musisz się nimi przejmować).

5. Załóżmy, że rejestr $a0 przechowuje wartość zmiennopozycyjną zapisaną w standardzie IEEE 754 pojedynczej precyzji. Napisz program, który mnoży tę wartość przez 2^k, gdzie k jest zawartością rejestru $a1. Wynik ma być zapisany również w tym samym standardzie, w rejestrze $v0. Nie wolno używać rozkazów koprocesora zmiennopozycyjnego. Zakładamy dla uproszczenia, że wyrzucamy z formatu IEEE 754 wszystkie „sytuacje wyjątkowe" (jak NaN, nieskończoności, czy zdenormalizowane mantysy).

6. (2 pkt.) Załóżmy, że rejestry $a0 i $a1 przechowują warości zmienno pozycyjne zapisane w standardzie IEEE 754 pojedynczej precyzji. Napisz program, który mnoży te wartości przez siebie i wynik umieszcze w rejestrze $v0. Tak jak w poprzednim zadaniu nie wolno używać rozkazów koprocesora zmiennopozycyjnego. I również zakładamy, że wyrzucamy z formatu IEEE 754 wszystkie „sytuacje wyjątkowe".

7. Przygotuj program sprawdzający czy MARS zachowuje się jak maszyna „little-endian", czy „big-endian", tzn. w jakiej kolejności w słowie pamięci przechowuje kolejne bajty 4-bajtowej liczby całkowitej.

8. Napisz funkcję otrzymującą w rejestrze $a0 adres napisu i zmieniajcą w tym napisie duże litery na małe, a małe na duże (zakładamy, ze w napisie będą tylko litery alfabetu łacińskiego i spacje). Przetestują tą funkcję wczytując dane z klawiatury i wyświetlając wynik na ekran (za pomocą wywołań odpowiednich funkcji systemowych).

9. W asemblerze procesora MIPS napisz program wyznaczający rekurencyjnie wartość n-tego wyrazu ciągu Fibonacciego. Liczbę n wczytaj z klawiatury.