Architektury systemów komputerowych

Lista 9
$x_9 = 4$ (minimum na bdb)

Rozwiązania tych zadań będziemy przedstawiać podczas ćwiczeń na tablicy (chociaż oczywiście zalecam przetestowanie ich np. przy użyciu SPIMA). Wkrótce opublikuję listę zadań (i zasady ich zaliczania), które będą oceniane przy komputerze.

1. Załóżmy, że rejestr $s1 przechowuje przechowuje wartość zmiennopozycyjną zapisaną w standardzie IEEE 754. Napisz program, który mnoży tę wartość przez 2^k, gdzie k jest zawartością rejestru $s2. Wynik ma być zapisany również w tym samym standardzie, w rejestrze $s3. Nie wolno używać rozkazów koprocesora zmiennopozycyjnego. Postaraj się wykrywać błąd nadmiaru zmiennopozycyjnego. Zakładamy dla uproszczenia, że wyrzucamy z formatu IEEE 754 wszystkie „sytuacje wyjątkowe".

2. Sprawdź na swoim komputerze, czy SPIM zachowuje się jak maszyna „little-endian", czy „big-endian" tzn. w jakiej kolejności w słowie pamięci przechowuje kolejne bajty 4-bajtowej liczby całkowitej. Na tablicy należy przedstawić program, którego użyłeś oraz wyniki jego działania. Spróbuj też umieścić w pamięci jakiś tekst za pomocą dyrektywy `.asciiz` i zobacz w jakiej kolejności ona umieści w pamięci znaki.

3. W asemblerze procesora MIPS napisz program, który wczytuje za pomocą funkcji systemowych łańcuch znaków, a następnie wypisuje go na wyjście z zamienionymi dużymi literami na małe, a małymi na duże (zakładamy, ze we wczytanym napisie będą tylko litery alfabetu łacińskiego i spacje).

4. W asemblerze procesora MIPS napisz program wyznaczający rekurencyjnie wartość n-tego wyrazu ciągu Fibonacciego.

5. W asemblerze procesora MIPS napisz rekurencyjną funkcję obliczającą współczynniki dwumianowe Newtona („n po k") zgodnie z definicją:

   $Newton(n,k) = 1$, jeśli $k = 0$ lub $k = n$
   $Newton(n,k) = Newton(n-1,k-1) + Newton(n-1,k)$ w pozostałych przypadkach.

   Przeanalizuj jej działanie dla wywołania z parametrami $n = 4$, $k = 2$. Zwróć uwagę przede wszystkim na zawartość stosu wywołań.

6. Rozważmy następujący kod w języku C:

```
int count (int a[], int n, int x){
   int res=0;
   int i;
   for (i=0; i!=n; i++)
      if (a[i]==x)
         res=res+1;
   return res;
}
```

   (a) Przetłumacz kod na język asmeblera MIPS.

   (b) Przepisz go na kod w C z użyciem wskaźników zamiast tablic (por. slajd 22 z wykładu 9),

   (c) Przetłumacz kod z punktu b) na język asemblera MIPS.

7. * Na wykładzie rzuciliśmy okiem na listę rozkazów procesora ARM. Zwróciliśmy uwagę na dwa rozwiązania odróżniające ARM od MIPS: skomplikowane tryby adresowania oraz możliwość warunkowego wykonywania każdej instrukcji (na podstawie specjalnych bitów stanu Z,N,C,O). Przedstaw przykład małego programu (sam wymyśl co program ma robić) w asemblerze ARM,

który wykorzystuje te rozwiązania. Ten sam program przepisz do języka asemblera MIPS. Porównaj kody obydwu programów.

Uwaga. W zadaniu tym chodzi właściwie o mały referat, w którym, przed prezentacją programu, należy opowiedzieć jak używa się odpowiednich trybów adresowania i kodów warunkowych w asemblerze. Osoba prezentująca rozwiązanie zadania przy tablicy może dodatkowo dostać 1-2 punkty za „jakość prezentacji".

Emanuel Kieroński