計算機システム Ⅱ ・中野・水 1 講時

2020 年 1 月 29 日

I 次の 2 つのプログラムは、ある C プログラムと、それを Intel 64 アーキテクチャの 64 bit モードの機械語プログラムに変換した結果である。ただし、関数 `main` に対応する部分の機械語プログラムは `0x400b5c` 番地から始まっている。また、関数呼び出しの際には、第 1 引数の値はレジスタ `rdi` に、第 2 引数の値はレジスタ `rsi` に格納し、戻り値はレジスタ `rax` に格納した状態で呼び出し元に戻るようにしている。この機械語プログラムを (関数 `main` を呼び出して) 実行するものとして、以下の問いに答えなさい。(28 点)

```
int *p;
int x;

void foo(int *q, int a) {
    if (*q [ >=  ] a) {
        *q = a;
    }
}

int bar(int n) {
    return n + [ 10 ];
}

int main() {
    x = 55;
    p = &x;
    foo(p, bar(50));
    return x;
}
```

```
0x400b4d:  movl   (%rdi), %eax
0x400b4f:  cmpl   %eax, %esi
0x400b51:  jg     0x400b55
0x400b53:  movl   %esi, (%rdi)
0x400b55:  retq
0x400b56:  movl   %edi, %eax
0x400b58:  addl   $0xa, %eax
0x400b5b:  retq
0x400b5c:  movl   $0x37, 0x6bc3a8
0x400b67:  movq   $0x6bc3a8, 0x6bc3a0
0x400b73:  movq   0x6bc3a0, %rbx
0x400b7b:  movl   $0x32, %edi
0x400b80:  callq  0x400b56
0x400b85:  movq   %rbx, %rdi
0x400b88:  movl   %eax, %esi
0x400b8a:  callq  0x400b4d
0x400b8f:  movl   0x6bc3a8, %eax
0x400b96:  retq
```

(1) 2 つのプログラムが対応するように空欄を埋めなさい。

(2) 変数 `p` の値を記憶しているメモリアドレスの範囲を 16 進数表記で答えなさい。

0x6bc3a0 〜 0x6bc3a7

(3) `0x400b4f` 番地の `cmpl` 命令が実行されるときのレジスタ `eax` の値を 16 進数表記で答えなさい。

0x37

(4) `0x400b51` 番地の `jg` 命令の次に実行される機械語命令のアドレスを 16 進数表記で答えなさい。

0x400b55

(5) `0x400b80` 番地の `callq` 命令が実行される際に、スタックにプッシュされる値 (リターンアドレス) を 16 進数表記で答えなさい。

0x400b85

(6) `0x400b8a` 番地の `callq` 命令が実行されるときのレジスタ `rbx` の値を 16 進数表記で答えなさい。

0x6bc3a8

学籍番号　　氏名

(裏面に続く)

II 情報実習室の Linux 環境で、下の C プログラムの空欄部分を右の A、B、C のいずれかで埋めて関数 foo を定義したプログラムを 3 通り作成した。関数 foo を呼び出して、その実行に必要となる時間を測定したところ、実行時間が最も短いものから順に、0.14 秒、0.20 秒、6.7 秒という結果となった。A、B、C それぞれは、どの実行時間に対応していると考えられるか、理由とともに 200 ～ 300 字程度で説明しなさい。(21 点)

A
```
sum += a[i][i];
```

B
```
sum += a[j][j];
```

C
```
sum += a[i][j];
```

```
#define N (20000)

int a[N][N];

int foo(void)
{
    int i, j;
    int sum = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            [          ]

        }
    }
    return sum;
}
```

int 型のデータの大きさが 4 byte である場合、内側の for 文で j が 1 だけ大きくなると、配列 a の要素にアクセスするときの番地は、A では 0 番地、B では 80004 番地、C では 4 番地ずれることになる。配列 a 全体の大きさは 400MB となり、キャッシュメモリには収まらないと考えられるため、キャッシュミスの頻度は、B が最も大きく、ついで C、A の順となると考えられる。よって、実行時間もこの順に長くなり、B が 6.7 秒、C が 0.20 秒、A が 0.14 秒であると考えられる。

(次ページに続く)

III 情報実習室の Linux 環境にログインし、端末エミュレータ (新しい端末) を開き、下図右のような C プログラム bar.c を、下図左のようにコンパイルして実行した。下の候補の中から最も適当と思われる語句を選んで空欄を埋めなさい。ただし、下図右の C プログラム中の行頭の数字は行番号であり、このプログラムの一部ではない。(30 点)

```
t180000@s01cd0542-160:~$ cc -o bar bar.c
t180000@s01cd0542-160:~$ ./bar
データ? 23
データ? 12
データ? 43
データ? 67
データ? 42
データ? -1
最大値 = 67
t180000@s01cd0542-160:~$
```

bar.c

```
 1 #include <stdio.h>
 2
 3 int main() {
 4     int x;
 5     int max = -1;
 6
 7     while (1) {
 8         printf("データ? ");
 9         scanf("%d", &x);
10         if (x < 0)
11             break;
12         if (x > max)
13             max = x;
14     }
15     if (max >= 0)
16         printf("最大値 = %d\n", max);
17     return 0;
18 }
```

(1) [シェル] はキーボードから「./bar」という文字列を読み取ると、[fork] (あるいはそれに類する) システムコールを利用して自らのプロセスを複製して新しいプロセスを生成し、その仮想アドレス空間に execve を使って bar の機械語プログラムなどを取り込み、その実行を開始する。

(2) この「./bar」は相対パス名と呼ばれる書き方である。カレントディレクトリの絶対パス名が /home/t190000 である場合、同じファイルを [/home/t190000/bar] という絶対パス名で指し示すことができる。

(3) bar のプロセスが起動された後、C プログラムの 9 行目で呼び出されている関数 scanf は、[read] というシステムコールを使って、標準入力 (ファイル記述子 [0] 番) から入力された文字列を読み取り、十進表記の整数値と解釈して、引数 &x で指定されたアドレスにその整数値を書き込む。このアドレスは、OS (カーネル) のメモリ管理機能によってプロセスごとに用意された [仮想] アドレス空間におけるアドレスである。

(5) bar.c の 4 ～ 5 行目に宣言されている変数 x や max は、bar のプロセスのメモリ空間内の [スタック] 領域と呼ばれる部分に割り当てられる。

(6) Linux などで採用されている [プリエンプティブ] マルチタスク方式の OS では、ユーザープロセスが CPU の [非特権] モードで実行されている状態でも、[タイマー] からの割り込み信号を受けると、CPU が [特権] モードに移行し、カーネルがあらかじめ設定しておいたプログラムが実行されることでプロセスのスケジューリングの処理を行うことができる。

空欄の候補

bar、../bar、/home/bar、/home/./bar、/home/t190000/bar、/home/t190000/bar.c、0、1、2、3、4、BSS、close、execve、fork、main、open、printf、read、scanf、write、アイドル、アドレス、イベント、カーネル、カレント、キーボード、キャッシュ、シェル、スタック、タイマー、テキスト、データ、ディレクトリ、パイプライン、ヒープ、ファイル、ブートストラップ、プリエンプティブ、プロセス、メモリ、ルート、レジスタ、親、階層的、仮想、協調的、子、実行可能、数理、絶対、相対、特権、非特権、物理

学籍番号　　氏名

(裏面に続く)

IV デマンドページング方式の仮想記憶システムを持ったオペレーティングシステムが稼働しており、あるプロセスが実行されている。このプロセスが使用できる (主記憶装置の) 物理ページは常に 6 ページであり、ページフォルトが起こると、(必要なら) 最後にアクセスされた時刻が最も遠い過去であるような物理ページを (補助記憶装置に) ページアウトし、(必要なら) アクセスしたい仮想ページの内容を (補助記憶装置から) その物理ページにページインする。(21 点)

(1) このプロセスがある時点までにアクセスした仮想ページの番号をアクセスした順に並べると次のようになった。

5, 0, 3, 3, 8, 1, 8, 5, 6, 4, 3, 3, 7, 9, 2, 1, 4, 4, 6, 2, 8, 8, 6, 4, 7, 4, 3, 8, 4, 4

この時点で (6 つの) 物理ページに割り当てられている仮想ページの番号をすべて挙げなさい。

最後にアクセスされた時刻が現在に最も近い 6 ページが物理ページに対応するため、物理ページに割り当てられている仮想ページの番号は

2, 3, 4, 6, 7, 8

の 5 つとなる。

(2) (1) に続いて、このプロセスが

5, 3, 2, 5, 1, 0, 7, 2

の順に、ここに挙げた番号の仮想ページにアクセスするとする。これら 8 回のアクセスでは、いつページフォルトが起こると考えられるか。ページフォルトが起こる仮想ページの番号を、ページフォルトが起こる順に列挙しなさい。

(1) の状況で、最後のアクセスが新しい順に仮想ページ番号を並べると、4, 8, 3, 7, 6, 2 となっている。この状態で 5 番の仮想ページへアクセスすると、このページは物理ページに対応していないのでページフォルトが発生し、2 番がページアウトし、5 番がページインする。以下、同様に考えて、ページフォルトが起こる順にページ番号を列挙すると

5, 2, 1, 0, 7

となる。

(3) このプロセスが、0 番から 19 番までの 20 個の仮想ページの内の 1 つをランダムに選んでアクセスすることを繰り返すとすると、1 回のアクセスでページフォルトが発生する確率はどの程度と考えられるか。

20 個の仮想ページの内、6 個が物理ページに対応した状態となるため、求める確率は

$$\frac{20-6}{20} = \frac{7}{10}$$

程度と考えられる。

(定期試験問題終り)