計算機システム II (第 15 回 期末試験問題) ・中野・水 1 講時

2021 年 1 月 13 日

注意: 解答はすべて別紙の解答用紙に記入すること。

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 `0x400b8d` 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。(36 点)

```
int foo(int x)
{
    if (x < 0)
        x = -x;
    return x;
}

void bar(int *p, int y)
{
    if (*p  (a)  foo(y))
         (b)  = *p + 10;
}

int a;

int main()
{
    a = 113;
    bar(  (c)  , -100);
    return a;
}
```

```
400b6d:    movl   %edi, %eax
400b6f:    cmpl   $0x0, %eax
400b72:     (d)   0x400b76
400b74:    negl   %eax
400b76:    retq
400b77:    movq   %rdi, %rbx
400b7a:    movl   %esi, %edi
400b7c:    callq  0x400b6d
400b81:    movl   (%rbx), %ecx
400b83:    cmpl   %ecx, %eax
400b85:    jle    0x400b8c
400b87:    addl   $0xa, %ecx
400b8a:    movl   %ecx, (%rbx)
400b8c:    retq
400b8d:    movl   $113, 0x6bc38c
400b98:    movq   $0x6bc38c, %rdi
400b9f:    movl   $-100, %esi
400ba4:    callq  0x400b77
400ba9:    movl   0x6bc38c, %eax
400bb0:    retq
```

(1) 2 つのプログラムが対応しているとすると、(a) ～ (d) の空欄はそれぞれどのようなものとなるか。

(2) 関数 `bar` の機械語プログラムの開始アドレスを 16 進数表記で答えなさい。

(3) 変数 `a` 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。

(4) `0x400b76` 番地の `retq` 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。

(5) `0x400b83` 番地の `cmpl` 命令が実行されるときのレジスタ `ecx` の値を 16 進数表記で答えなさい。

(6) `0x400ba4` 番地の `callq` 命令が実行されるとき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。

(次ページに続く)

II 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、右のような C プログラム baz.c を、下のようにコンパイルして実行した。空欄を埋めるべきものとして、最も適当と思われる語句を下の候補の中から選ん答えなさい。ただし、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。(39 点)

baz.c

```
 1 #include <stdio.h>
 2
 3 #define SIZE (1000)
 4
 5 int a[SIZE];
 6
 7 int sort(int d[], int n)
 8 {
 9     int i, j, tmp;
10
11     for (i = 0; i < n - 1; i++) {
12         for (j = i + 1; j < n; j++) {
13             if (d[j] < d[i]) {
14                 tmp = d[i];
15                 d[i] = d[j];
16                 d[j] = tmp;
17             }
18         }
19     }
20 }
21
22 int main()
23 {
24     int i, n = 0;
25
26     printf("data = ");
27     while (n < SIZE) {
28         if (scanf("%d", &a[n]) != 1)
29             break;
30         n++;
31     }
32     sort(a, n);
33     printf("result = ");
34     for (i = 0; i < n; i++)
35         printf("%d ", a[i]);
36     printf("\n");
37     return 0;
38 }
```

```
t190000@s01cd0542-160:~$ pwd
/home/t190000
t190000@s01cd0542-160:~$ cd prog
t190000@s01cd0542-160:~/prog$ cc -o baz ../baz.c
t190000@s01cd0542-160:~/prog$ ./baz
data = 75 32 56 23 11 92 27 71 23 4 66.
result = 4 11 23 23 27 32 56 66 71 75 92
t190000@s01cd0542-160:~/prog$
```

キーボードから入力された「pwd」や「cd prog」という文字列を読み取っているのは一般に [(1)] と呼ばれるプログラムのプロセスである。このプロセスは標準入力 (ファイル記述子 [(2)] 番) から [(3)] というシステムコールを使ってこれらの文字列を読み込むが、キーボードから文字列が入力されるまでの間は、このプロセスは [(4)] 待ち状態となっており、カーネルは CPU 資源を他のユーザープロセスの実行に割り当てることができる。

この実行例では「baz.c」というソースファイルをコンパイルしているが、このファイルは [(5)] という絶対パス名で指し示すこともできる。

実行例中で baz というオブジェクトプログラムを起動している部分では、シェルは、まず、[(6)] というシステムコールによって、自分自身のプロセスを複製し、次に、[(7)] というシステムコールによって、新しいプロセスの仮想アドレス空間に baz の機械語プログラムなどを取り込み、特定の番地からその実行を開始する。

起動された baz のプロセスは、標準出力 (ファイル記述子 [(8)] 番) に「data = 」という文字列を出力しているが、これは標準入出力ライブラリ関数の一つである [(9)] の機械語プログラムが、[(10)] というシステムコールを呼び出すことで行なわれている。一方、標準入力から、「75 32 56 23 11 92 27 71 23 4 66.」という文字列を読み込む際には、同じく標準入出力ライブラリ関数の一つである [(11)] の機械語プログラムが read というシステムコールを呼び出す。

配列 a はこのプロセスの仮想メモリ空間の [(12)] 領域に、一方、関数 main 内で宣言されている変数 n は、[(13)] 領域に割り当てられているはずである。

空欄の候補

prog/baz.c、../prog/baz.c、../baz.c、baz.c、/prog/baz.c、/home/t190000/baz.c、/home/t190000/prog/baz.c、-1、0、1、2、3、4、BSS、close、execve、fork、main、open、printf、read、scanf、sort、unlink, write、アイドル、アドレス、イベント、カーネル、カレント、キャッシュ、シェル、スタック、テキスト、ディレクトリ、デバイス、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、実行、数理、絶対、相対、特権、非特権、物理

(次ページに続く)

III 4 MiB の主記憶装置 (物理メモリ) を持つ計算機システムで、1 ページの大きさが 2 KiB であるような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、右の C プログラムをコンパイルして得られる機械語プログラムのプロセスが実行されている。このプロセスでは、配列 a のための領域として、常に (4 MiB の内の) 4000 KiB (2000 ページ) の物理メモリを占有できている。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。int 型のデータの大きさは 4 B で、配列 a はある仮想ページの先頭アドレスから割り当てられているものとして次の問いに答えなさい。ただし、キャッシュメモリの影響は考えないものとする。(25 点)

prog.c

```
 1 #define NUM 1500
 2 #define STEP 64
 3 #define LEN (NUM*1024)
 4
 5 int a[LEN];
 6
 7 int main()
 8 {
 9     int i, sum = 0;
10
11     while (1) {
12         for (i = 0; i < LEN; i += STEP) {
13             sum += a[i];
14         }
15     }
16     return 0;
17 }
```

(1) 配列 a 全体を記憶するために必要とされる仮想ページの数を求めなさい。

(2) プログラム 13 行目の「`sum += a[i];`」の部分で配列 a の要素へアクセスする際に、ページフォルトが起こる確率 (割合) はどの程度と考えられるか。

(3) NUM の値は変えずに、STEP の値の定義を、128、256、512、1024、2048、... のように、2 倍ずつ大きくしていくと、「`sum += a[i];`」でページフォルトが (当該ページへの最初のアクセスを除いて) 起こらなくなるのは、STEP の値をいくつに定義したときか。

(4) 逆に、STEP の値は 64 に固定しておき、NUM の値を減らしていくとすると、どこまで減らしたときに、「`sum += a[i];`」部分でページフォルトが (当該ページへの最初のアクセスを除いて) 起こらないようになるか。

(期末試験問題終り)

計算機システム Ⅱ (第 15 回 期末試験) 解答用紙

Ⅰ (1) (a) `<` (b) `*p` (c) `&a` (d) `jge`

(2) 400b77

(3) 6bc38c 〜 6bc38f

(4) 400b81

(5) 71

(6) 400ba9

Ⅱ (1) シェル

(2) `0`

(3) `read`

(4) イベント

(5) `/home/t190000/baz.c`

(6) `fork`

(7) `execve`

(8) `1`

(9) `printf`

(10) `write`

(11) `scanf`

(12) `BSS`

(13) スタック

学籍番号　　氏名

(裏面に続く)

Ⅲ (1)

$(4 \times 1500 \times 1024)/(2 \times 1024) = 3000$ より、3000 ページ

(2)

12 行目の `for` 文では配列 `a` 全体にアクセスすることになるが、配列 `a` は 3000 ページ分であるから、2000 ページの物理メモリに収まらない。`for` 文が繰り返されるたびに、配列 `a` の添字は 64 ずつ増えるので、アクセスするアドレスは 256 B 分ずつずれることになる。1 ページは 2048 B なので、$2048/256 = 8$ 回アクセスする度に次のページへ移ることになる。このときアクセスされるページは最も古い過去にアクセスしたページとなるから、ここでページフォルトが起こるが、続く 7 回のアクセスは同じページ中のアドレスとなるのでページフォルトは起きない。よって、求める確率は $\frac{1}{8}$ となる。

(3)

13 行目でアクセスされるアドレスのずれがページサイズの 2 倍になれば、この `for` 文の繰り返しでアクセスされるページの総数は配列全体の大きさである 3000 ページの半分となり、2000 ページの物理メモリに収めることができ、ページフォルトは起らなくなる。よって、求める `STEP` の大きさは $(2 \times 2048)/4 = 1024$ となる。

(4)

この場合、配列 `a` のすべてのページに順にアクセスすることになるので、配列 `a` 全体が 2000 ページの物理メモリに収まる必要がある。よって、$4 \times \texttt{NUM} \times 1024 = 2000 \times 2048$ を解いて、$\texttt{NUM} = 1000$ となる。

(解答用紙終わり)