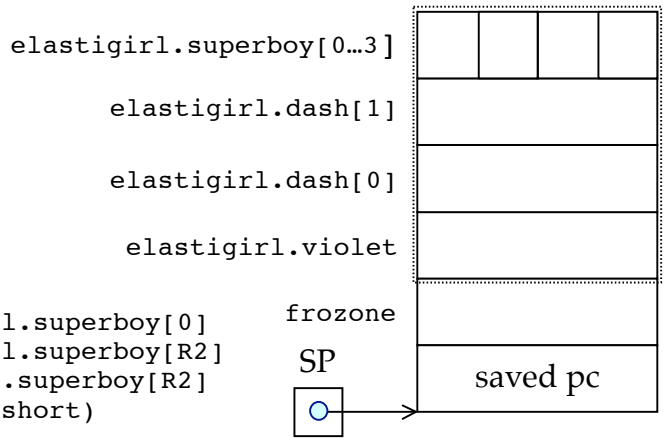
Section Solution

Problem 1: The Incredibles

Consider the following **`struct`** definition:

```
typedef struct {
   int violet;
   char *dash[2];
   char superboy[4];
} superhero;

static superhero **pixar(superhero *syndrome);
static superhero *theincredibles(short *frozone, superhero elastigirl)
{
line 1    frozone += elastigirl.superboy[*frozone];
line 2    ((superhero *)((superhero *) elastigirl.dash[0])->dash)->violet = 400;
line 3    return *pixar(&elastigirl) + 10;
}
```



Generate code for the entire **`theincredibles`** function. Be clear about what assembly code corresponds to what line.

```
// line 1
R1 = M[SP + 4];   // R1 stores frozone
R2 =.2 M[R1];     // R2 stores *frozone
R3 = SP + 20;     // R3 stores &elastigirl.superboy[0]
R4 = R3 + R2;     // R4 stores &elastigirl.superboy[R2]
R5 =.1 M[R4];     // R5 stores elastigirl.superboy[R2]
R6 = R5 * 2;      // scale rhs by sizeof(short)
R7 = R1 + R6;     // compute result
M[SP + 4] = R7;   // store result back to frozone

// line 2
R1 = M[SP + 12];  // load elastigirl.dash[0];
M[R1 + 4] = 400;  // load 400 into the violet field of the struct we
                  // pretend starts at R1 + 4 (&dash[0] is 4 bytes above
                  // bottom of struct)

// line 3
R1 = SP + 8;      // prepare argument
SP = SP – 4;      // make space for superhero * on stack
M[SP] = R1;       // initialize parameter
CALL <pixar>;     // jump to code
SP = SP + 4;      // clean up params, RV contains superhero **
R2 = M[RV];       // R2 contains superhero *
RV = R2 + 160;    // RV should be set 10 * sizeof(superhero) beyond that

// end of function, no locals, RV is set
RET;
```

Problem 2: New Zoo Revue (given on a midterm several years ago)

You are to generate code for the following nonsense code. Don't be concerned about optimizing your instructions or conserving registers. We don't ask that you draw the activation records, but it can only help you get the correct answer if you do. Be clear about what assembly code corresponds to each line of C.

```
   struct human {
      int doug;
      short emmyjo[2];
   };

   struct other {
      char *freddie;
      struct human charlie;
      struct human *henrietta;
   };

   static struct human **AskingQuestions(struct human *heroes);
   static struct human **BeingCalm(short *conformity, struct other *gooddeeds)
   {
line 1    conformity[*conformity] = 0;
line 2    gooddeeds += ((struct human *) (gooddeeds->henrietta[0].emmyjo))->doug;
line 3    return AskingQuestions((struct human *) &gooddeeds);
   }
```

- Generate code for the entire **`BeingCalm`** function.

```
// line 1
R1 = M[SP + 4];   // load conformity
R2 =.2 M[R1];     // load *conformity
R2 = R2 * 2;      // manually compute offset from base address
R3 = R1 + R2;     // manually compute address
M[R3] =.2 0;      // and write a short 0 there..

// line 2
R1 = M[SP + 8];   // load gooddeeds, remember it
R2 = M[R1 + 12];  // load henrietta[0] field
R3 = M[R2 + 4];   // load doug field of human struct at emmyjo
R4 = R3 * 16;     // pointer math on struct other *: quantum is 16 bytes
R5 = R1 + R4;     // compute new value for gooddeeds
M[SP + 8] = R5;   // flush back to stack

// line 3
R1 = SP + 8;      // remember &gooddeeds
SP = SP – 4;      // make space for param
M[SP] = R1;
CALL <AskingQuestions>
SP = SP + 4;      // clean up space

RET;              // leave RV alone, return
```

- The assembly code generated by the compilation of **`AskingQuestions`** is given below. Provide a C-code implementation of **`AskingQuestions`**, given the constraint that all local variables must be of type **`struct human`** and/or **`struct human *`**, and the only typecast allowed is **`(struct other *)`**.

```
SP = SP – 12;
R1 =.2 M[SP + 8];
M[SP + 4] = R1;
R2 = M[SP + 16];
BNE R2, 0, PC + 16; // potentially jumps to instruction loading RV
R3 = M[SP];
R4 = R3 + 8;
M[SP] = R4;
RV = SP + 12;
SP = SP + 12;
RET;
```

Answer

```
static struct human **AskingQuestions(struct human *heroes)
{
   struct human manners;
   struct human *remainingCalm;
   manners.doug = manners.emmyjo[0];
   if (heroes == NULL) remainingCalm++;
   return &remainingCalm + 3;
}
```

The **`struct human`** is 8 bytes, and pointers are 4 bytes. The **`SP = SP – 12`** line makes it clear that there are either two or three local variables—one pointer and one direct structure, or three pointers. The **`.2`** implies that a **`short`** resides at byte offset 8, so that further requires that the first variable be of type **`struct human`**, and therefore the second variable is a pointer.