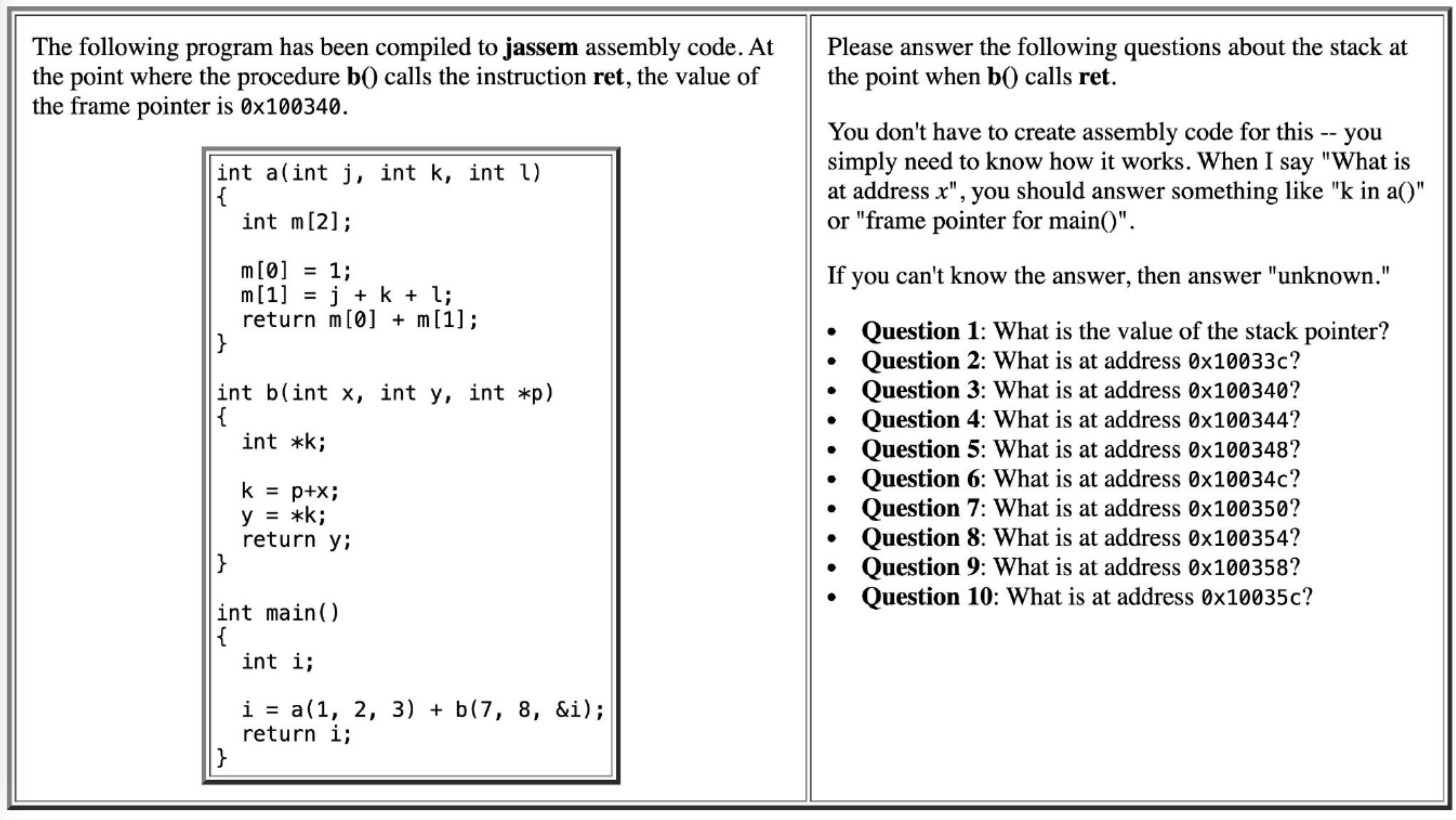


The following program has been compiled to **jassem** assembly code. At the point where the procedure **b()** calls the instruction **ret**, the value of the frame pointer is `0x100340`.

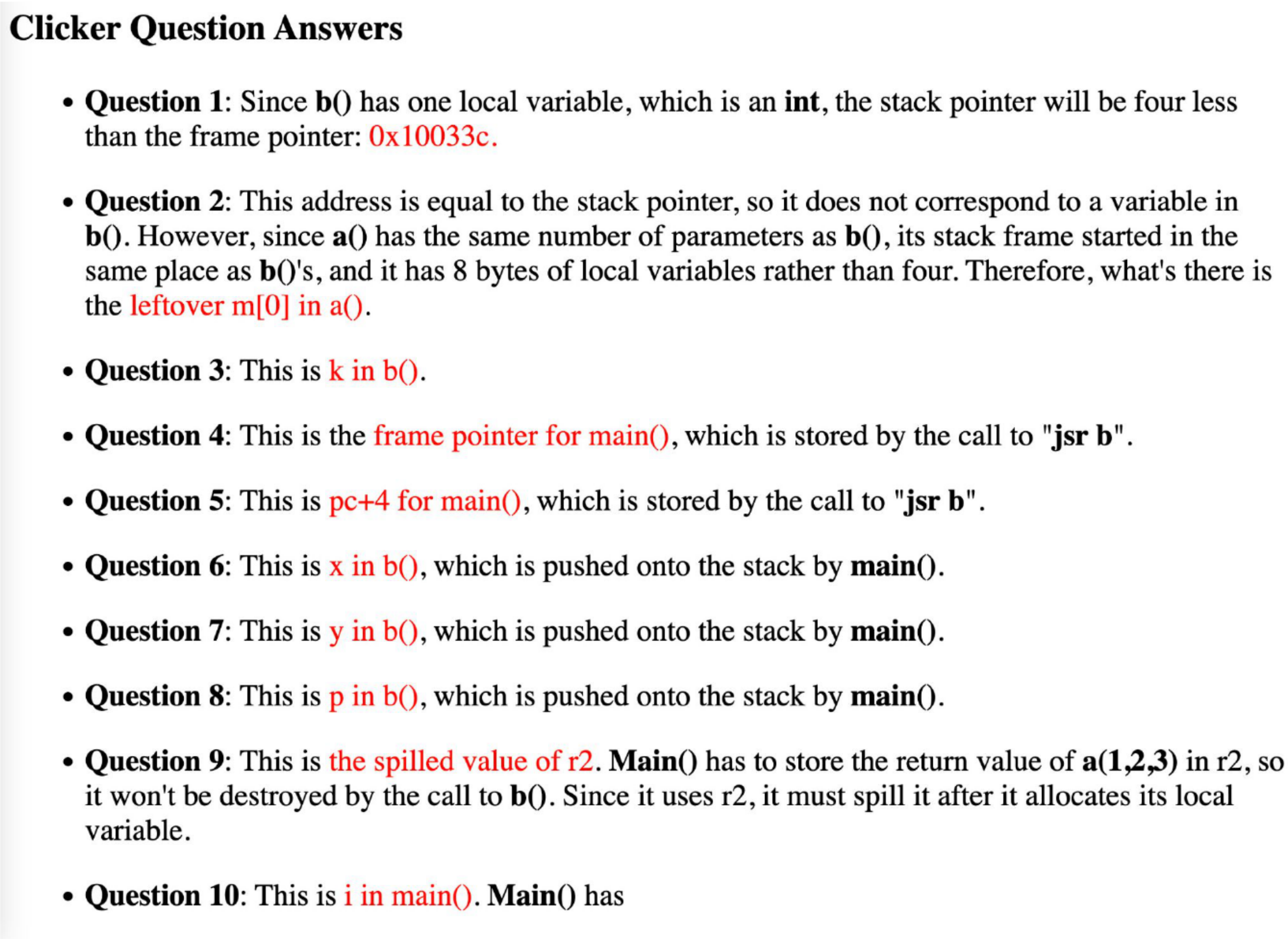
```
int a(int j, int k, int l)
{
  int m[2];

  m[0] = 1;
  m[1] = j + k + l;
  return m[0] + m[1];
}

int b(int x, int y, int *p)
{
  int *k;

  k = p+x;
  y = *k;
  return y;
}

int main()
{
  int i;

  i = a(1, 2, 3) + b(7, 8, &i);
  return i;
}
```

Please answer the following questions about the stack at the point when **b()** calls **ret**.

You don't have to create assembly code for this -- you simply need to know how it works. When I say "What is at address *x*", you should answer something like "k in a()" or "frame pointer for main()".

If you can't know the answer, then answer "unknown."

- **Question 1**: What is the value of the stack pointer?
- **Question 2**: What is at address `0x10033c`?
- **Question 3**: What is at address `0x100340`?
- **Question 4**: What is at address `0x100344`?
- **Question 5**: What is at address `0x100348`?
- **Question 6**: What is at address `0x10034c`?
- **Question 7**: What is at address `0x100350`?
- **Question 8**: What is at address `0x100354`?
- **Question 9**: What is at address `0x100358`?
- **Question 10**: What is at address `0x10035c`?



Clicker Question Answers

- **Question 1**: Since **b()** has one local variable, which is an **int**, the stack pointer will be four less than the frame pointer: 0x10033c.
- **Question 2**: This address is equal to the stack pointer, so it does not correspond to a variable in **b()**. However, since **a()** has the same number of parameters as **b()**, its stack frame started in the same place as **b()**'s, and it has 8 bytes of local variables rather than four. Therefore, what's there is the leftover m[0] in a().
- **Question 3**: This is k in b().
- **Question 4**: This is the frame pointer for main(), which is stored by the call to "**jsr b**".
- **Question 5**: This is pc+4 for main(), which is stored by the call to "**jsr b**".
- **Question 6**: This is x in b(), which is pushed onto the stack by **main()**.
- **Question 7**: This is y in b(), which is pushed onto the stack by **main()**.
- **Question 8**: This is p in b(), which is pushed onto the stack by **main()**.
- **Question 9**: This is the spilled value of r2. **Main()** has to store the return value of **a(1,2,3)** in r2, so it won't be destroyed by the call to **b()**. Since it uses r2, it must spill it after it allocates its local variable.
- **Question 10**: This is i in main(). **Main()** has