1) int arith(int x, int y)

```
        pushl %ebp
        movl %esp,%ebp
        movl 12(%ebp),%eax
        movl 8(%ebp),%edx
        subl %eax,%edx
        imull %eax,%edx
        movl %edx,%eax
        movl %ebp,%esp
        popl %ebp
        ret
```

```
int arith(int x, int y)
{
  int result = 0;
  result = (x-y)*y;
  return result;
}
```

- subl is tricky
- result is kept in %edx and not allocated on stack
- result = 0 was removed by compiler

2) int mem(int *xp, int y)

```
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%eax
        movl 12(%ebp),%edx
        movl (%eax),%ecx
        movl %edx,(%eax)
        movl %ecx,%eax
        movl %ebp,%esp
        popl %ebp
        ret
```

```
int mem(int *xp, int y)
{
  int result;

  result = *xp;
  *xp = y;

  return result;
}
```

- result is kept in %ecx

3) int optarith(int x, int y)

```
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%eax
        sall $4,%eax
        movl %eax,%edx
        subl 8(%ebp),%edx
        movl 12(%ebp),%eax
        testl %eax,%eax
        jge .L6
        addl $3,%eax
.L6:
        sarl $2,%eax
        addl %eax,%edx
        movl %edx,%eax
        movl %ebp,%esp
        popl %ebp
        ret
```

```
int optarith(int x, int y)
{
  int result = 0;

  result = x*15 + y/4;

  return result;
}
```

- x*15 is implemented as x*16 (shift) - x
- division is implemented as right shift, correction for negative numbers

4) int ifcode(int x, int y)

```
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%edx
        testl %edx,%edx
        jge .L8
        movl $1,%eax
        movl %edx,%ecx
        cltd
        idivl %ecx
        jmp .L9
        .p2align 4,,7
.L8:
        movl 12(%ebp),%eax
.L9:
        movl %ebp,%esp
        popl %ebp
        ret
```

```
int ifcode(int x, int y)
{
  int result = 0;

  result = x < 0 ? 1/x : y;

  return result;
}
```

- idiv <opd>:
  %eax = (%edx:%eax)/<opd>;
  $edx = (%edx:%eax)%<opd>
- cltd: sign extends %eax to %edx

5) int whilecode(int x, int y)

```
        pushl %ebp
        movl %esp,%ebp
        pushl %esi
        pushl %ebx
        movl 8(%ebp),%ecx
        movl 12(%ebp),%esi
        xorl %ebx,%ebx
        testl %ecx,%ecx
        je .L12
        .p2align 4,,7
.L13:
        movl %ecx,%edx
        sarl $31,%edx
        movl %esi,%eax
        andl %edx,%eax
        addl %eax,%ebx
        addl %ecx,%ecx
        jne .L13
.L12:
        movl %ebx,%eax
        popl %ebx
        popl %esi
        movl %ebp,%esp
        popl %ebp
        ret
```

```
int whilecode(int x, int y)
{
  int result = 0;

  while (x) {
    int mask = (x >> 31);
    result += y&mask;
    x <<= 1;
  }

  return result;
}
```

- note how while loops are implemented
- loop variable is in %ecx
- result is accumulated in %ebx
- xorl %ebx, %ebx -> %ebx = 0
- left shift by one is implemented as add

6) int forcode(int x, int y)

```
        pushl %ebp
        movl %esp,%ebp
        pushl %edi
        pushl %esi
        pushl %ebx
        movl 8(%ebp),%ebx
        movl 12(%ebp),%edi
        xorl %edx,%edx
        xorl %ecx,%ecx
        cmpl %ebx,%edx
        jge .L17
        movl $1,%esi
.L19:
        movl %esi,%eax
        sall %cl,%eax
        andl %edi,%eax
        addl %eax,%edx
        incl %ecx
        cmpl %ebx,%ecx
        jge .L17
        cmpl $31,%ecx
        jle .L19
.L17:
        movl %edx,%eax
        popl %ebx
        popl %esi
        popl %edi
        movl %ebp,%esp
```

```
popl %ebp
ret
```

```
int forcode(int x, int y)
{
  int result = 0;
  int i;

  for (i = 0; i < x && i < 32; i++)
    result += y & (1 << i);

  return result;
}
```

- loop variable is in %ecx
- result is accumulated in %edx
- %cl (lower two bytes of %ecx) is used for shifting