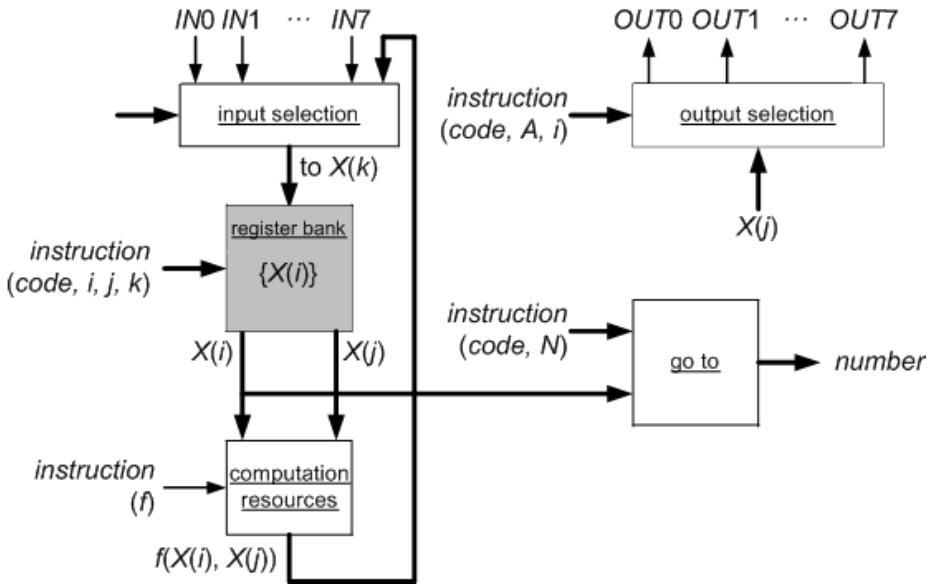
P3.1 STRUCTURAL SPECIFICATION
(continuation)

Jean-Pierre Deschamps

University Rovira i Virgili, Tarragona, Spain

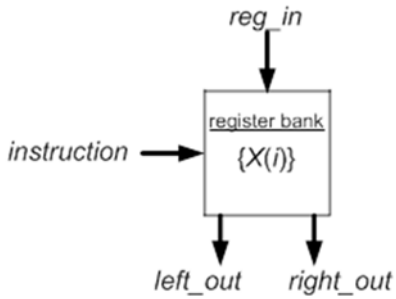
2 BLOCK DESCRIPTION

2.3 REGISTER BANK




```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

(register bank)



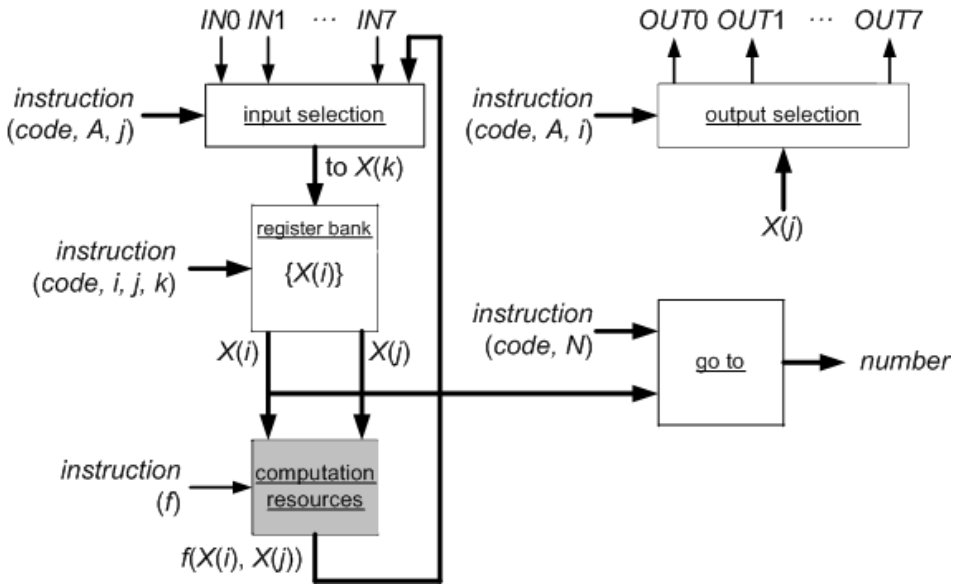
Functional specification

```
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := reg_in; left_out := don't care; right_out := don't care;
    when (DATA_INPUT, k, j) =>
      X(k) := reg_in; left_out := don't care; right_out := don't care;
    when (DATA_OUTPUT, i, j) =>
      left_out := don't care; right_out := X(j);
    when (OUTPUT_VALUE, i, A) =>
      left_out := don't care; right_out := don't care;
    when (OPERATION, i, j, k, f) =>
      X(k) := reg_in; left_out := X(i); right_out := X(j);
    when (JUMP, N) =>
      left_out := don't care; right_out := don't care;
    when (JUMP_POS, i, N) =>
      left_out := X(i); right_out := don't care;
    when (JUMP_NEG, i, N) =>
      left_out := X(i); right_out := don't care;
  end case;
end loop;
```



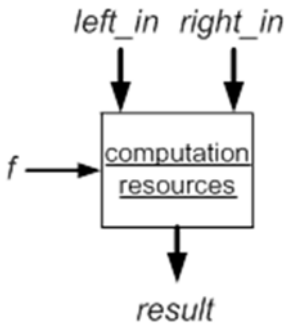
```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

2.4 COMPUTATION RESOURCES




```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

(computation resources)



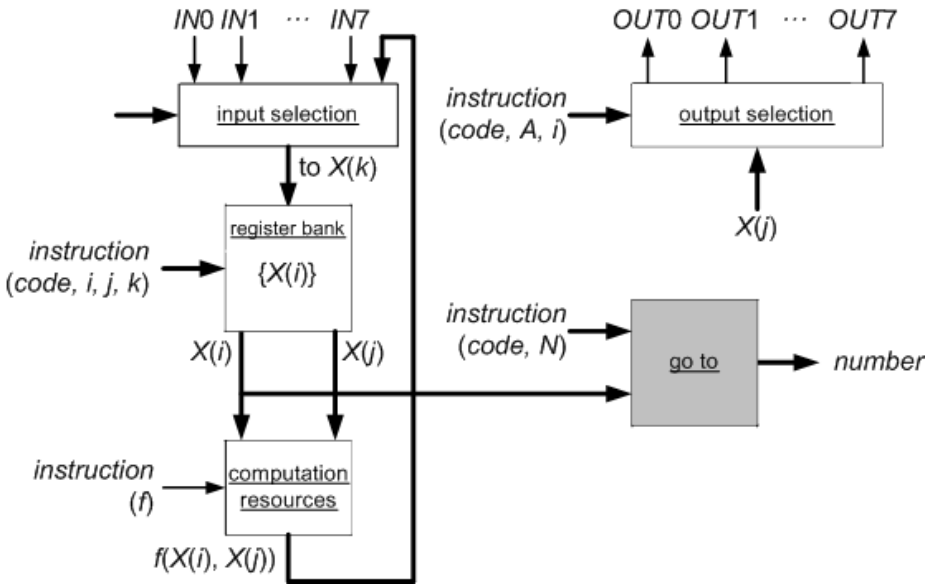
Functional specification

if f = 0 then result := left_in + right_in;
else result := left_in - right_in;
end if;

```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

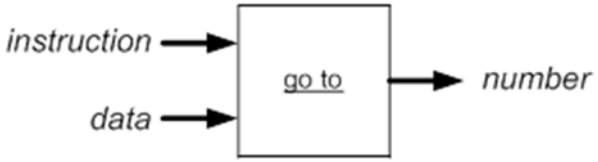
UAB Universitat Autònoma de Barcelona

2.5 GO TO




```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

(go to)




```
number := 0;
loop
  case program(number) is
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if data > 0 then number := N;
      else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if data < 0 then number := N;
      else number := number + 1; end if;
    when others =>
      number := number + 1;
  end case;
end loop;
```

```
number := 0;
loop
  case program(number) is
    when (ASSIGN_VALUE, k, A) =>
      X(k) := A; number := number + 1;
    when (DATA_INPUT, k, j) =>
      X(k) := IN(j); number := number + 1;
    when (DATA_OUTPUT, i, j) =>
      OUT(i) := X(j); number := number + 1;
    when (OUTPUT_VALUE, i, A) =>
      OUT(i) := A; number := number + 1;
    when (OPERATION, i, j, k, f) =>
      X(k) := f(X(i), X(j)); number := number + 1;
    when (JUMP, N) =>
      number := N;
    when (JUMP_POS, i, N) =>
      if X(i) > 0 then number := N; else number := number + 1; end if;
    when (JUMP_NEG, i, N) =>
      if X(i) < 0 then number := N; else number := number + 1; end if;
  end case;
end loop;
```

SUMMARY

P3.1

Universitat Autònoma de Barcelona

- **Structural description** completed.

- **Functional description of blocks**:

✓ register bank,
✓ computation resources,
✓ go to block.