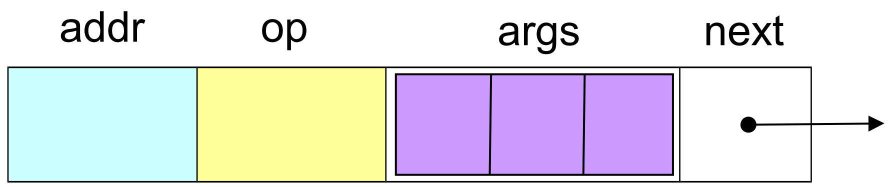
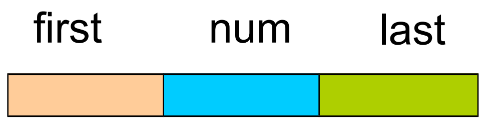
Strutture Dati per la Generazione di Codice

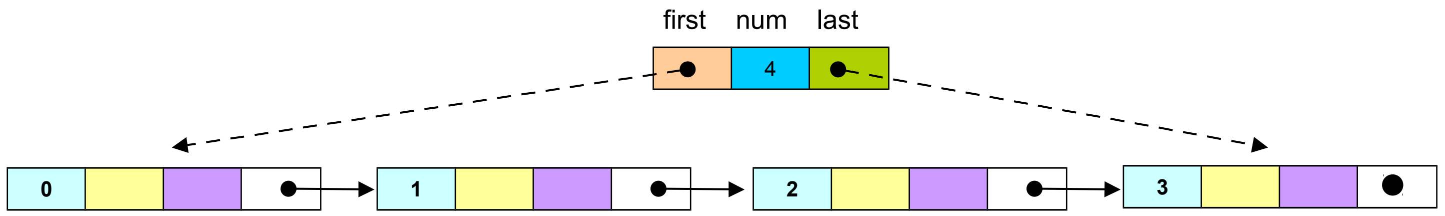
- `Stat:`



- `Code:`



- Rappresentazione di un segmento di codice (sequenza di istruzioni S-code):



Libreria di Funzioni per Generazione di Codice

```
void relocate_address(Code code, int offset)

Code appcode(Code code1, Code code2)

Code endcode()

Code concode(Code code1, Code code2, ...)

Stat *newstat(Operator op)

Code makecode(Operator op)

Code makecode1(Operator op, int arg)

Code makecode2(Operator op, int arg1, int arg2)

Code make_psh_pop(int num_formals, int num_variables, int entry)

Code make_lci(int i)

Code make_lcr(float r)

Code make_lcs(char *s)
```

relocate_address()

```
void relocate_address(Code code, int offset)
{
  Stat *p = code.first;
  int i;

  for(i = 1; i <= code.num; i++)
  {
    p->addr += offset;
    p = p->next;
  }
}
```

appcode()

```
Code appcode(Code code1, Code code2)
{
  Code rescode;

  relocate_address(code2, code1.num);
  rescode.first = code1.first;
  rescode.last = code2.last;
  code1.last->next = code2.first;
  rescode.num = code1.num + code2.num;
  return rescode;
}
```

endcode(), concode()

```
Code endcode()
{
  static Code code = {NULL, 0, NULL};

  return code;
}

Code concode(Code code1, Code code2, ...)
{
  Code rescode = code1, *pcode = &code2;

  while(pcode->first != NULL)
  {
    rescode = appcode(rescode, *pcode);
    pcode++;
  }
  return rescode;
}
```

newstat(), makecode(), makecode1(), makecode2()

```c
Stat *newstat(Operator op)
{
  Stat *pstat;

  pstat = (Stat*) malloc(sizeof(Stat));
  pstat->addr = 0;
  pstat->op = op;
  pstat->next = NULL;
  return pstat;
}
```

```c
Code makecode(Operator op)
{
  Code code;

  code.first = code.last = newstat(op);
  code.num = 1;
  return code;
}
```

```c
Code makecode1(Operator op, int arg)
{
  Code code;

  code = makecode(op);
  code.first->args[0].ival = arg;
  return code;
}
```

```c
Code makecode2(Operator op, int arg1, int arg2)
{
  Code code;

  code = makecode1(op, arg1);
  code.first->args[1].ival = arg2;
  return code;
}
```

make_psh_pop()

```
Code make_psh_pop(int num_formals, int num_variables, int entry)
{
    return concode(makecode2(PSH, num_formals, num_variables),
                   makecode1(GOT, entry),
                   makecode(POP),
                   endcode());
}
```

make_lci(), make_lcr(), make_lcs()

```
Code make_lci(int i)
{
   return makecode1(LCI, i);
}
```

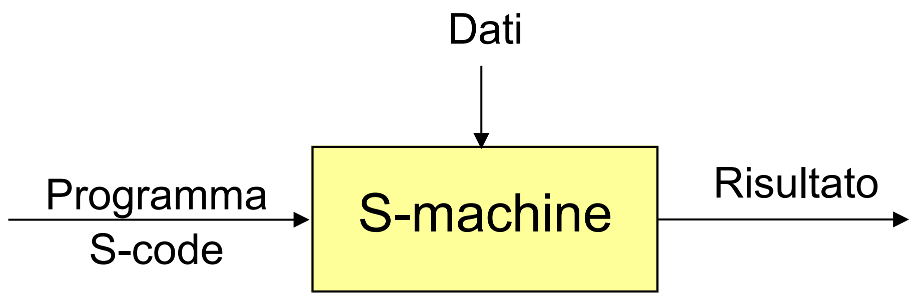
```
Code make_lcr(float r)
{
   Code code;

   code = makecode(LCR);
   code.first->args[0].rval = r;
   return code;
}
```

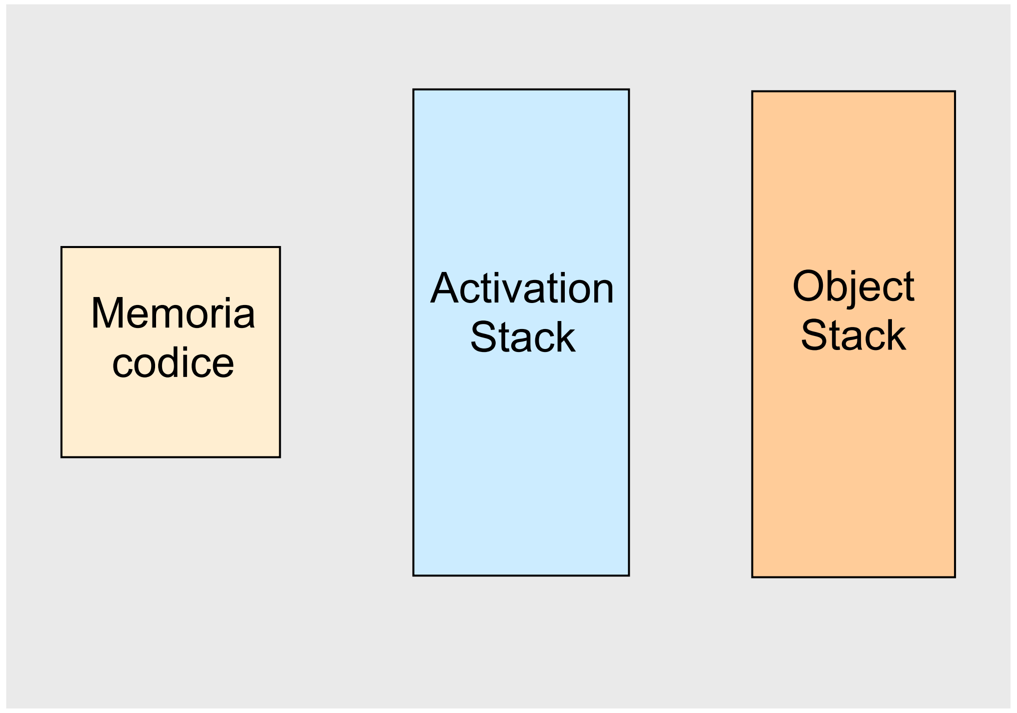
```
Code make_lcs(char *s)
{
   Code code;

   code = makecode(LCS);
   code.first->args[0].sval = s;
   return code;
}
```

Macchina Astratta

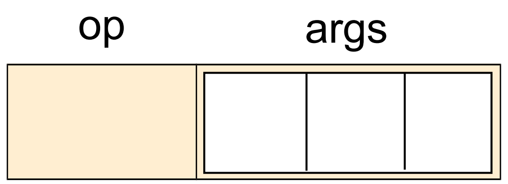


- **Architettura:**



Memoria Codice

- `Scode`:



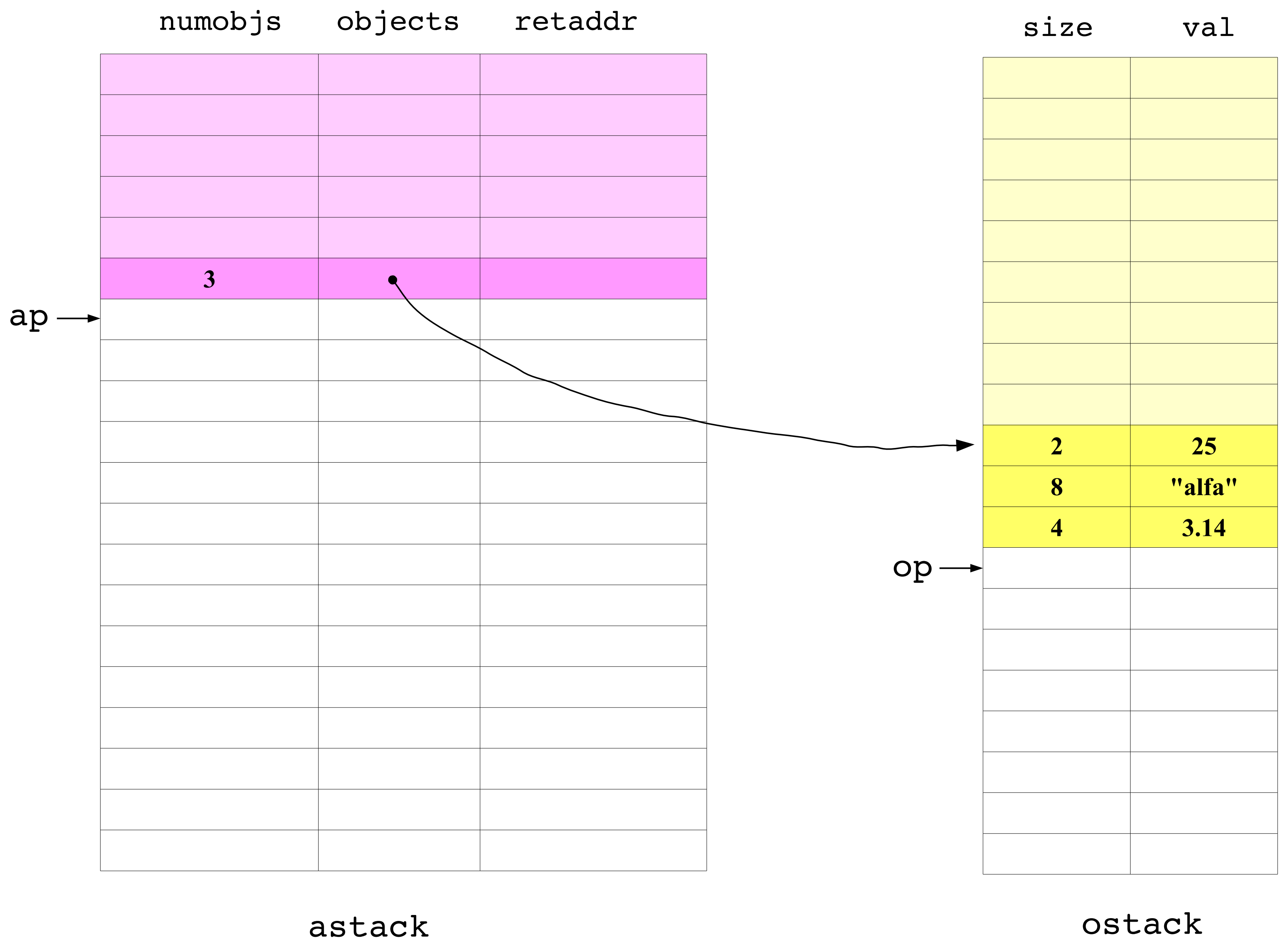
```
#define MAXARGS 3

typedef struct
{
    Operator op;
    Lexval args[MAXARGS];
} Scode;

Scode *prog;
```

- Allocata nella inizializzazione della S-machine $\rightarrow$ `SIZ` *size*

Stack di Attivazione e Stack degli Oggetti


numobjs
objects
retaddr
3
ap
astack
size
val
2
25
8
"alfa"
4
3.14
op
ostack