Selection

```
selector(struct item_t* item) {
 struct object_t* object;
 struct item_t* indexItem;

 while ((symbol == ARROW) || (symbol == LBRAK))
  if (symbol == ARROW) {
   getSymbol();

   if (symbol == IDENTIFIER) {
    object = findObject(item->type->fields, identifier);

    if (object != NULL)
     field(item, object);
    else
     error("unknown field: ", identifier);

    getSymbol();
   } else
    error("identifier expected");
  } else if (symbol == LBRAK) {
   getSymbol();
   indexItem = malloc(sizeof(struct item_t));
   expression(indexItem);

   index(item, indexItem);

   if (symbol == RBRAK)
    getSymbol();
   else
    error("missing ']'");
  }
}
```

dereferences record and memorizes field offset

dereferences array and computes index offset

takes us into REF mode

→ we need to remain in REF mode since item may appear in left hand side of assignment!

Field and Index

```
field(struct item_t* item, struct object_t* object) {
 // unlike with Oberon loading is necessary here for dereferencing
 load(item);

 item->mode = REF_MODE;
 item->offset = object->offset;
 item->type = object->type;
}

ref2Reg(struct item_t* item) {
 item->mode = REG_MODE;

 put(LDW, item->reg, item->reg, item->offset);

 item->offset = 0;
}

index(struct item_t* item, struct item_t* indexItem) {
 if (indexItem->mode == CONST_MODE) {
  // unlike with Oberon loading is necessary here for dereferencing
  load(item);

  item->mode = REF_MODE;
  item->offset = indexItem->value * 4;
 } else {
  load(indexItem);

  put(MULI, indexItem->reg, indexItem->reg, 4);

  // unlike with Oberon loading is necessary here for dereferencing
  load(item);

  item->mode = REF_MODE;
  put(ADD, item->reg, item->reg, indexItem->reg);
  releaseRegister(indexItem->reg);
 }

 item->type = item->type->base;
}
```

may be in VAR or REF mode

dereferences array or record reference!

no new register needed (unlike from VAR mode)

we only use types of size 32 bits here

reduces # of registers used at the same time by one!

next are examples!